Dynamic Similarity Metric for Case-Based Planning Under Imperfection

BY

MALIK AHMED OWAIS

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

May 2002

UMI Number: 1419522

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 1419522

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by

Malik Ahmed Owais

under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Thesis Committee

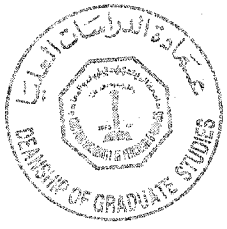
Dr. Moataz A. Ahmed (Chairman)

Dr. Muhammad Sarfraz (Member)

Dr. Muhammed Shafique (Member)

Department Chairman

Prof. Osama A. Jannadi
(Dean of Graduate Studies)



16/6/2012
Date

To My Parents

ACKNOWLEDGEMENTS

All praise to Allah, the most Beneficent and the most Merciful, who enabled me to complete my thesis work. I make a humble effort to thank Allah for his endless blessings on me, as His infinite blessings cannot be thanked for. Then, I pray Allah to bestow peace on his last prophet Muhammad (*Sal-allah-'Alaihe-Wa-Sallam*) and on all his righteous followers till the day of judgement.

I am deeply grateful to my thesis advisor Dr. Moataz A. Ahmed for his valuable guidance and cooperation throughout my thesis work. At the same time, gratitude is due to my thesis committee members Dr. Muhammad Sarfraz and Dr. Muhammed Shafique. I express my thanks to all of them for their valuable time and support.

I acknowledge the academic and computing facilities provided by the Information and Computer Science Department of King Fahd University of Petroleum and Minerals (KFUPM).

I pay a heartily tribute to all of my family members and especially to my parents, who guided me during all my life endeavors. Their love and support motivated me to continue my education and achieve higher academic goals. Without their moral support and sincere prayers, I would have been unable to accomplish this task.

Finally, I appreciate the friendly support from all my colleagues at KFUPM. In particular, I want to thank Asif, Faisal, Shafayat, Akhtar and Rufai.

Contents

List of Tables

List of Figures

THESIS ABSTRACT

Name: Malik Ahmed Owais

Title: Dynamic Similarity Metric for Case-Based Planning Under Imperfection

Major Field: Computer Science

Date of Degree: May 2002

Planning and reasoning have been a major area of research in Artificial Intelligence (AI). Case-based planning (CBP) is one of the knowledge-based planning technique, which develops new plans by using its past experience instead of planning from scratch. CBP caches the solutions to planning problems in a case base *and provides methods for retrieving those solutions, whose problem descriptions are similar to a new given problem. The task of retrieving similar cases (problem-solution pairs) becomes difficult when the environment is imprecise and uncertain, this makes the whole CBP process complex. In this work, we propose a similarity metric for retrieval step of CBP under imperfect environment. For capturing the imprecise information, we use fuzzy predicates to represent the initial and goal states of a case and a new problem. The similarity between a problem and a case, computed by fuzzy predicate based similarity metric (FDSM), is dynamic i.e. similarity may change after problem solving episodes have been completed. This is because of weights assigned to the predicates of a case and a new problem predicates. These weights are updated/learned on the basis of a feedback from plan repairer. In this work, for simulation purpose, a fuzzy plan replay system is developed for computing the plan repair effort.*

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum and Minerals, Dhahran.

May 2002

خلاصة الرسالة

الاسم : مالك أحمد عويس

العنوان: مقياس التشابه الديناميكي لتخطيط قاعدة الحالة تحت النقص
الدرجة: الماجستير في العلوم
التخصص الرئيسي: علوم الحاسب الآلي
تاريخ التخرج: مايو 2002

لقد كان التخطيط والتفكر أهم مجالان للبحث والدراسة في الذكاء الاصطناعي. حيث تعتبر تخطيط قاعدة الحالة (سي بي بي) أحد تقنية تخطيط قاعدة المعرفه (knowledge-based)، الذي يطور الخطط الجديده باستعمال تجاربها السابقة بدلاً من أن تخطط من البداية. يخبئ سي بي بي الحلول لتخطيط المشاكل في قاعدة حالة ثم يزود الأساليب لاسترجاع تلك الحلول، الذي أوصاف مشاكلهم مشابهة لمشكله معطية جديدة. إن مهمة استرجاع حالات التشابه (هيئة أزواج حل- مشكلة) تصبح عملية صعبة عندما تكون البيئة المحيطة غير دقيقة ومجهولة ، فتجعل كل عملية سي بي بي معقدة. في هذا العمل ، نقترح مقياس التشابه لخطوة الاسترجاع لسي بي بي تحت البيئة الناقصة. ولاقتناص المعلومات الغير دقيقة ، نستعمل مسندات الغائم (fuzzy predicates) لتمثيل الوضع الإبتدائي والوضع النهائي (الهدف) لحالة أو مشكلة جديدة. يعتبر التشابه بين المشكلة والحالة – محسوبة باستخدام مقياس التشابه المبني على مسندات الغائم – ديناميكي أي أن التشابه قد يتغير بعد إكمال وقائع حل المشكلة. السبب في ذلك هي الأوزان التي تسند لمسندات حالة ولمسندات مشكلة جديدة. وتُجدد / تُعلم هذه الأوزان على أساس رد من مصلّح الخطة. في هذا العمل ومن أجل المحاكاة ، تم تطوير نظام إعادة خطة الغائم لحساب جهد تصليح الخطة.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن
مايو 2002

Chapter 1

Introduction

Reasoning about actions and plans is fundamental to the development of intelligent machines, which are capable of dealing effectively with real world problems. The need to plan sequence of actions to achieve goals arises in many applications; e.g. robotics, scheduling scientific experiments, managing scarce resources, sequencing in the shipping industry etc.

Planning is the problem of devising a series of actions that will lead to a desired goal. The task of a planner is to find a sequence of actions that allows accomplishment of some specific task. The roots of Artificial Intelligence (AI) planning lie partly in problem solving through state-space search and associated techniques such as problem reduction and means-ends analysis, and partly in theorem proving and situation calculus.

A *plan* is an organized collection of operators or actions. A plan is said to be a solution to a given problem if the plan is applicable in the problem's initial state,

and if after plan execution, the goal is true. It means that all the preconditions of any action, within the plan, are satisfied before applying that action. A *state* is a collection of characteristics of an object that is sufficiently detailed to uniquely determine the new characteristics of the object that will result after an action. The *initial state* description tells the planning system the way the world is "right now". The *goal situation(s)* description tells the planning system the way we want the world to look when the plan has been executed. The world, in which planning takes place, is often called the *application domain.* In many planning systems, a goal may be transformed into a set of other, usually simpler, goals called "subgoals". The classical approach, that most planners use today, describes states and operators in the terminology of STRIPS [1].

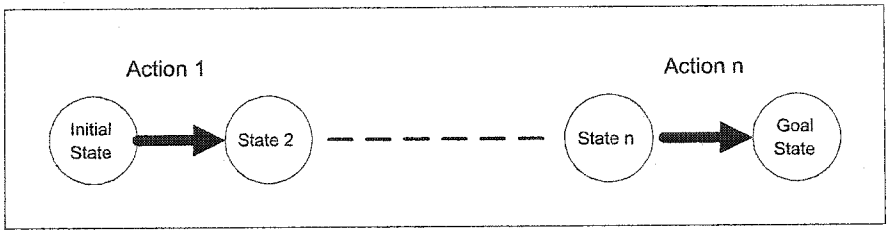


Figure 1.1: Plan: sequence of actions.

1.1 Key Ideas behind planning

Problem solving is the process of building a system to solve a particular problem, given in the form of goal situations(s) and initial state(s). One way of achieving this is simply to explore the possible steps that can be taken in the initial state,

and then move on to do the same thing with all the states that are achieved by one of the possible steps. This is done by applying the operators to the current state, thereby generating a new set of states. This process is called expanding the state. To solve the problem we simply build a tree, with the aim of finding a sequence of branches (steps) which connects the initial state with the goal state. Some other approaches for problem-solving can be found in [2] and [3].

Planning can be viewed as a type of problem solving in which the planner uses beliefs about actions and their consequences to search for a solution over space of plans or over the space of situations. But a planner differs from a problem solver in representation of goals, states and actions, and in the representation and construction of action sequences. In problem solving, description of the initial state is given, and the only information about the goal is in the form of goal test and heuristic function, which are used as black boxes. Also during the construction of solutions, search algorithms consider only unbroken sequences of actions beginning from the initial state. The fact that the problem-solver considers sequence of actions starting from the initial state also contributes to its difficulties. These limitations of problem-solver motivate the design of planning systems. The following are the key ideas behind planning [4]:

- The first key idea behind planning is to "open up" the representation of states, goals, and actions. States and goals are represented by sets of sentences, and actions are represented by logical descriptions of preconditions and effects.

This enables the planner to make direct connections between states and actions.

- The second key idea behind planning is that *the planner is free to add actions to the plan whenever they are needed, rather than in an incremental sequence starting at the initial state.* There is no necessary connection between the order of planning and the order of execution. By making obvious or important decisions first, the planner can reduce branching factor for future choices and reduce the need to backtrack over arbitrary decisions.

- The third key idea behind planning is that most parts of the world are independent of most other parts. So divide-and-conquer algorithm can be used the independent subproblems, but it fails if the cost of combining the solutions to subproblems is too high.

1.2 Plan Representation

A plan is formally defined as a data structure consisting of the following four components [4]:

- A set of plan steps. Each step is one of the available operators for solving the problem.

- A set of step ordering constraints. Each ordering constraint is of the form $S_i \prec S_j$,which is read as "S_i before S_j" and means that step S_i must occur

sometime before S_j (but not necessarily immediately before).

- A set of variable binding constraints. Each variable constraint is of the form $v = x$, where v is a variable in some step, and x is either a constant or another variable.

- A set of causal links. A causal link is written as $S_i \xrightarrow{c} S_j$ and read as "S_i achieves c for S_j." Causal links serve to record purpose(s) of steps in the plan: here a purpose of S_i is to achieve the precondition c of S_j.

1.3 Approaches to planning

Broadly, there are four distinct approaches to planning. They are

- Nonhierarchical planning
- Hierarchical planning /Abstraction level planning
- Script-based planning
- Opportunistic planning

Nonhierarchical planning corresponds roughly to the colloquial meaning of planning; that is, a nonhierarchical planner develops a sequence of actions to achieve each of its goals. It may reduce goals to simpler ones, or it may use means-end-analysis to reduce the differences between the current state of the world and the state that would hold after the plan has been made. The major disadvantage of nonhierarchical

planning is that it does not distinguish between actions that are critical to success of a plan and those that are simply details. Examples of nonhierarchical planners are STRIPS [1], HACKER [5] and INTERPLAN [6].

Hierarchical planning provides a means of ignoring the details that obscure or complicate solution to a problem. In hierarchical planning, first a plan is sketched that is complete but too vague and then the vague parts are refined into more detailed subplans until finally the plan has been refined to a complete sequence of detailed planning operators. The advantage of this approach is that the plan is first developed at a level at which the details are not computationally overwhelming. Examples of this approach are ABSTRIPS [7] and NOAH [8].

The distinction between the above two approaches is that hierarchical planners generate a hierarchy of representations of a plan in which the highest is a simplification, or abstraction, of the plan and the lowest is a detailed plan, sufficient to solve the problem. In contrast, nonhierarchical planners have only one representation of a plan.

In script-based planning, the skeleton of plan are recalled from a store of plans instead of generating them as in hierarchical planning. The planning process proceeds in two steps: First a skeleton plan is found that is applicable to the given problem and then the abstract steps in the plan are filled in with the planning operators from the particular problem context. Example of this approach is MOGLEN [9].

In opportunistic planning, a blackboard control structure has been developed to model human planning. The blackboard is a "clearinghouse" for suggestions about

plan steps. These suggestions are made by different specialists (planners or agents). Each specialist is designed to make a particular kind of planning decision. In this type of planning, the specialists do not operate in any particular order i.e. their coordination is asynchronous. In this approach, parts of plans can be developed independently, which makes it different from other planning approaches [10].

1.4 Knowledge-based Planners

Another way to classify the AI planners is according the search and knowledge-intensive paradigms.

Pure search-intensive methods search exhaustively for a solution from first principles,i.e., individual steps that model atomic actions in the task domain and may be chained to form a solution to problem. Pure knowledge-intensive methods for problem solving presuppose the existence of a collection of prototypical solutions from where the problem solver retrieves and instantiates an appropriate solution to a new problem. Variations from these two extreme approaches extend the search-intensive paradigm to search guided by local control knowledge while the knowledge-intensive extreme extends to case-based reasoning approaches in which the retrieved solution may be adapted after being retrieved and instantiated.

Knowledge-based planners are class of AI problem solving systems that are combination of both search-intensive and knowledge-intensive paradigms. Knowledge-based planners are broadly characterized by the fact that they apply domain knowledge to

search heuristically through a space of possible actions to find a sequence of actions that will achieve a goal. Within the AI community, there exist a number of competing paradigms for knowledge-based planning. Each paradigm provides general answers to questions such as the following: What is a plan? How should knowledge relevant to planning be represented? How should plans be generated and modified?

Examples of knowledge-based planning paradigm are: Goal-directed planning, Constraint-directed planning, Case-based planning, Transformational planning, planning as logical deduction, Reactive planning and Mathematical Optimization.[11].

1.5 Planning and imperfect information

In practical planning problems, it is rarely the case that all relevant information and outcomes are known with certainty and precision. Usually, there is substantial imperfection about both the initial state and the effects of proposed actions. Knowledge of a planner can be imperfect due to two reasons,

1. There may be doubt about validity of knowledge; it is then uncertain. This uncertainty arises because of
    - *laziness*: avoiding too much work to list complete set of rules.
    - *ignorance*: lack of complete theory or knowledge about a domain.
2. There may be some difficulty in expressing this knowledge precisely; it is then imprecise.

These two types of imperfection of knowledge are often closely combined. Thus, the real world appears both imprecise and uncertain.

In most applications, the planner does not have complete information about the state of the world. Planning for such applications involves figuring out what to do in the absence of the information. Sometimes we can plan to gather the necessary information. Other times we must proceed in ignorance making use of whatever information is available. Coping with incomplete information adds yet another dimension of computational complexity to planning.

In planning, different approaches have been used to deal with imperfection in knowledge, like fuzzy logic, probability theory, conditional planning etc.

In this thesis we address the utilization of one of the knowledge-based planning techniques, namely case-based planning (CBP), for solving the problems under imperfection. Case-based planning comprises of following main steps.

- Retrieval
- Adaptation and Evaluation
- Execution and Storage

In this thesis, particulary, we'll discuss the "retrieval" of similar cases in case-based planning under imperfection. We'll focus on how to improve the process of finding good analogs in imprecise environment because the obtainment of good analogs will reduce the need for adaptation.

We are utilizing fuzzy logic to represent imprecise knowledge in CBP. We have developed a fuzzy predicate-based similarity metric for matching the similar cases in retrieval step of case-based planning under imperfection. This similarity metric is dynamic i.e. it learns from previous case-based planning episodes.

1.6 Organization of the Thesis

The rest of the thesis is organized as follows; in chapter 2 we'll discuss different representation techniques for imperfection in planning, and issues about case-based planning, in particular. In chapter 3 different similarity measuring techniques will be discussed. Our solution approach will be described in chapter 4. Chapter 5 discusses results of simulations. The conclusion of this thesis as well as future work are presented in chapter 6.

Chapter 2

Background

In this chapter, we'll discuss different techniques which are used for handling imperfection in knowledge of a planner. Later we'll discuss case-based planning and its issues.

2.1 Handling imperfection

Following are some techniques which have been used for handling imperfection in knowledge of a planner.

2.1.1 Probability and Decision theory for uncertainty

One way to tackle uncertainty, which comes from laziness and ignorance, is through probability theory. The probability is the degree of belief. A probability of 0 for a given sentence (predicate) corresponds to an unequivocal belief that the sentence

is false, while the probability of 1 corresponds to an unequivocal belief that the sentence is true. Probabilities between 0 and 1 correspond to intermediate degrees of belief in the truth of the sentence. The sentence itself is in fact either true or false. A probability of 0.8 means 80% degree of belief- i.e. fairly strong expectation. If a planner assigns a probability of 0.8 to a sentence, then it expects that in 80% of cases that are indistinguishable from the current situation, the sentence will turn out to be true.

Decision theory [12] provides an attractive framework for weighting the strengths and weaknesses of a particular course of action, with roots in probability theory and utility theory. Utility theory is used for preferences between the possible outcomes of the various plans. The fundamental idea of decision theory is that a planner is rational if and only if it chooses the action that yields the highest expected utility averaged over all the possible outcomes of the action. This is called the principle of Maximum Expected Utility (MEU). Given a probability distribution over the possible outcomes of an action in any state, and a reasonable preference function over outcomes, we can define a utility function on outcomes such that whenever the agent would prefer one plan to another, the preferred plan has higher expected utility. The task of planner then seems straightforward - to find the plan with the MEU.

The recent increase in research in decision-theoretic planning follows the recent success of work in reasoning under uncertainty, and draws on compact representations such as those of belief nets [13]. Belief network is used to represent the

dependence between variables and to give a concise specification of the joint probability distribution.

2.1.2 Fuzzy sets and fuzzy Logic for imperfection

Fuzzy set theory is a means of specifying how well an object satisfies a vague description. *Fuzzy set theory* has at least two advantages;

- It allows imprecise knowledge to be modelled, such that an approximation of a numerical value ("about 3.80 meters"), or a vague information expressed, for instance in natural language ("high").
- It is the only way to deal, within one framework, with knowledge numerically provided by measuring devices, and knowledge symbolically expressed by a human observer for instance.

However, fuzzy set theory does not allow imprecision and uncertainty to be handled with the same formalism [14]. But these two types of imperfection in knowledge are closely linked, because in many cases the more accuracy is required for the terms of an assertion, the less certain becomes the assertion.

The following example of conversation illustrates this concept:

"How old is he?"

"About twenty"

"Are you sure?"

"Absolutely!"

"This is really imprecise!"

"Well, in fact, I think he is 31, but I am not sure."

Moreover, reasoning based on *imprecise knowledge* often generates *uncertainties* on the final decision that must be taken. The following reasoning illustrates this problem:

1. Being over 18, makes one eligible to vote.
2. We know that Peter is about 18, is he allowed voting?
3. It is quite *possible*, but *not certain*.

Possibility theory was introduced in 1978 by L.A. Zadeh [14] in connection with the fuzzy set theory, to allow a reasoning to be carried out on imprecise or vague knowledge, making it possible to deal with uncertainties on this knowledge. Possibility theory provides a method of formalizing non-probabilistic uncertainties on events, i.e. a means of assessing to what extent the occurrence of an event is possible and to what extent we are certain of its occurrence, without knowing the probability of this occurrence. The difference between probability and possibility theory is discussed at the end of this section.

Basic concepts of Fuzzy Sets

Fuzzy logic is a superset of the conventional (Boolean) logic that has been extended to handle the concept of partial truth – truth-values between "completely true" and "completely false". As there is a strong relationship between Boolean logic and the concept of a subset, there is a similar strong relationship between fuzzy logic and fuzzy subset theory.

In classical set theory, a subset U of a set S can be defined as a mapping from the elements of S to the elements of the set $\{0, 1\}$,

$$U : S \rightarrow \{0, 1\}$$

This mapping may be represented as a set of ordered pairs, with exactly one ordered pair present for each element of S. The first element of the ordered pair is an element of the set S, and the second element is an element of the set $\{0, 1\}$. The value zero is used to represent non-membership, and the value one is used to represent membership. The truth or falsity of the statement

x is in U

is determined by finding the ordered pair whose first element is x. The statement is true if the second element of the ordered pair is 1, and the statement is false if it is 0.

Similarly, a fuzzy subset F of a set S can be defined as a set of ordered pairs, each with the first element from S, and the second element from the interval [0,1], with exactly one ordered pair present for each element of S. This defines a mapping

between elements of the set S and values in the interval [0,1]. The value zero is used to represent complete non-membership, the value one is used to represent complete membership, and values in between are used to represent intermediate DEGREES OF MEMBERSHIP. The set S is referred to as the UNIVERSE OF DISCOURSE for the fuzzy subset F. Frequently, the mapping is described as a function, the MEMBERSHIP FUNCTION of F. The degree to which the statement

x is in F

is true is determined by the second element of the ordered pair whose first element is x. In practice, the terms "membership function" and fuzzy subset get used interchangeably.

If X is a collection of objects denoted generically by x, then a fuzzy set A in X is set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid \ x \in X\}$$

Fuzzy Predicates:

Predicates indicate relations between objects. In bivalent systems, represented by first or higher order logic, the predicates are crisp i.e. they can have only two values : either TRUE or FALSE. The examples of such predicates are *MORTAL(X), EVEN, LARGER THAN* etc, which can be either true or false. This is because *Truth* in bivalent systems can have only two values. In fuzzy logic, the predicates are fuzzy i.e. they are represented by fuzzy set and each predicate will have a degree of membership in a fuzzy set, e.g. *tall, soon, swift, slow, much larger than* etc. These predicates are called "fuzzy predicates".

In fuzzy logic *Truth* is treated as linguistic variable whose truth-values form a truth-values set such as shown below:

T(*Truth*) = {*true, not true, very true, completely true, more or less true, ..., false, very false, neither true nor false*}

The members of the truth-values set are ultimately represented by fuzzy sets. True(u) is a numeric truth value in the range [0,1] representing the truth of the proposition "u is true", which can be interpreted as the membership grade $\mu_{true}(u)$. For example, the membership of the numerical truth-value 0.7 with the linguistic truth-value *very true* might be 0.6.

If we consider SPEED as a linguistic variable then it can have the values *slow, fast, very fast* etc, and it can be represented by a fuzzy membership function. While the fuzzy predicates like *SLOW*(speed-value) or *SLOW*(SPEED(VEHICLE)), which represent how slow the speed is, will have truth values from truth values set T(*Truth*). So they can be *true, very true, more or less true, false, very false* etc. This is as opposed to the two valued logic predicate *SLOW*(speed-value), which is constrained to crisp in the sense that the denotation of a predicate must be a non-fuzzy subset of the universe of discourse i.e. either TRUE or FALSE.

In figure 2.1, fuzzy sets representing a possible vehicle-speed are shown. Also the speed of vehicle is shown as a fuzzy membership function with dotted lines.

Consider the example:

Old paintings are usually rare.

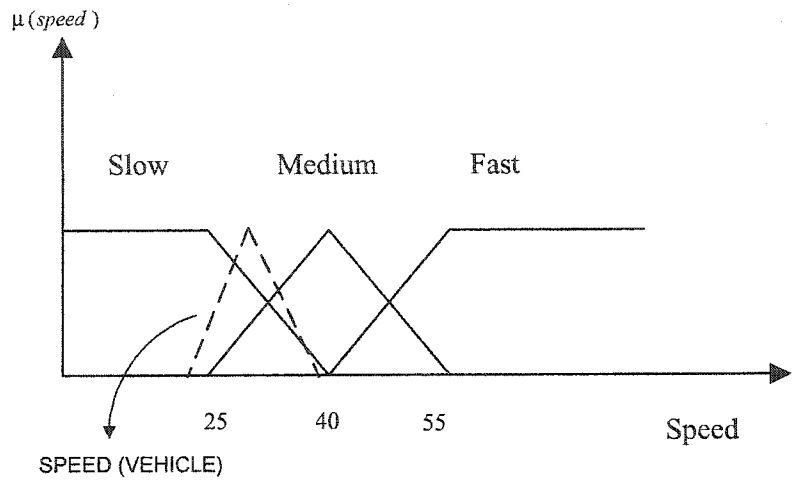


Figure 2.1: Fuzzy sets representing a possible vehicle-speed states

Rare items are expensive.

So Old paintings are usually expensive.

This fuzzy logic deduction depends on the fuzzy predicates: "expensive, old, rare" and fuzzy quantifiers: "usually". In Fuzzy predicate logic the above example can be expressed as

$$Old(Painting) \rightarrow UsuallyRare(Painting)$$

$$\forall item\ (Rare(item) \rightarrow Expensive(item))$$

Operators:

Each fuzzy predicate, in fuzzy logic, is represented by fuzzy set, but logical operators $\wedge, \vee, \rightarrow, \neg$ need redefinition for fuzzy values. The standard definitions of these operators in fuzzy logic are:

- *Membership function of the complement (representing $\neg$) $\bar{A}$ of a normalized*

fuzzy set A:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x), \qquad x \in X$$

- *Membership function of the intersection (representing $\wedge$)* $C = A \cap B$

$$\mu_C(x) = min\{\mu_A(x), \mu_B(x)\}, \qquad x \in X$$

- *Membership function of the union (representing $\vee$)* $C = A \cup B$

$$\mu_C(x) = max\{\mu_A(x), \mu_B(x)\}, \qquad x \in X$$

- The t-norm (triangular norm), denoted by $\underline{*}$, were proposed for fuzzy-sets intersection operation:

min {x,y}	*fuzzy intersection*
xy	*algebraic product*
$x \underline{*} y$ = max{0, x+y-1}	*bounded product*
x if y = 1; y if x = 1; 0 if $x, y < 1$	*drastic product*

- The t-conorm, denoted by $\underline{+}$, were proposed for fuzzy-sets union operation:

max {x,y}	*fuzzy union*
x + y - xy	*algebraic product*
$x \underline{+} y$ = min{1, x+y}	*bounded sum*
x if y = 0; y if x = 0; 1 if $x, y > 0$	*drastic sum*

- The Cartesian product $A \times B$ is a fuzzy set in $U \times U$ with the membership function defined for all $(u_1, u_2) \in U \times U$ by

$$\mu_{A \times B}(u_1, u_2) = min \mu_A(u_1, \mu_B(u_2$$

- Let A and B be fuzzy sets in U and V, respectively. A fuzzy implication, denoted by $A \rightarrow B$, is a fuzzy set in $U \times V$ with the following possible membership function:

$\mu_{A \rightarrow B}(u, v) = \mu_A(u) \underline{*} \mu_B(v)$ & *fuzzy conjunction*
$\mu_{A \rightarrow B}(u, v) = \mu_A(u) \underline{+} \mu_B(v)$ & *fuzzy disjunction*
$\mu_{A \rightarrow B}(u, v) = \mu_{\bar{A}}(u) \underline{+} \mu_B(v)$ & *material implication*
$\mu_{A \rightarrow B}(u, v) = \mu_{\bar{A}}(u) \underline{+} \mu_{A \underline{*} B}(v)$ & *propositional calculus*

Relationship between fuzzy truth-values and probabilities

In fact, from a mathematical perspective, fuzzy sets and probability exist as parts of a greater Generalized Information Theory that includes much formalism for representing uncertainty (including random sets, Dempster-Shafer evidence theory, probability intervals, possibility theory, general fuzzy measures, interval analysis, etc.). Semantically, the distinction between fuzzy logic and probability theory has to do with the difference between the notions of probability and a degree of membership. Probability statements are about the likelihood of outcomes: an event either occurs or does not, and you can bet on it. But with fuzziness, one cannot say unequivocally whether an event occurred or not, and instead you are trying to model the EXTENT to which an event occurred.

Comparison between Possibility and Probability

The properties of probability and possibility are different, but they are however comparable. In the same way, the properties of possibility distributions can be compared with the properties of probability distributions:

- *probability measures* summarize a body of precise and varied knowledge, while
- *possibility measures* reflect a vague but coherent knowledge.

The possibility functions are more natural for the representation of the feeling of incertitude: very precise information is not expected absolutely from someone, but we do hope for the greatest possible coherence in his remarks. On the other hand,

precise but fluctuating data more usually result from the observation of a physical phenomenon.

2.1.3 Dempster-Shafer Theory

The Dempster-Shafer theory [15] is designed to deal with the distinction between uncertainty and ignorance. Rather than computing the probability of a proposition, it computes the probability that the evidence supports the propositions. This measure of belief is called a belief function.

2.2 Case-Based Planning

Case-based planning (CBP) is the idea of planning as remembering. In this approach of planning, the past experiences are used in developing new plans, so planning is done from memory. A case-based planner has a library of cases, each case consisting of a problem description and its solution (plan).

A case-based planner uses its knowledge of the world and the effects of its actions in the world to build a plan. Plans, similar to the current situation, are retrieved from the memory and a new plan is constructed by modifying the most similar existing plan. This new plan is then stored in the memory. The modifications and additions change the planner's understanding of what can happen in different situations and make it possible for the planner to understand which plans are appropriate for certain situations.

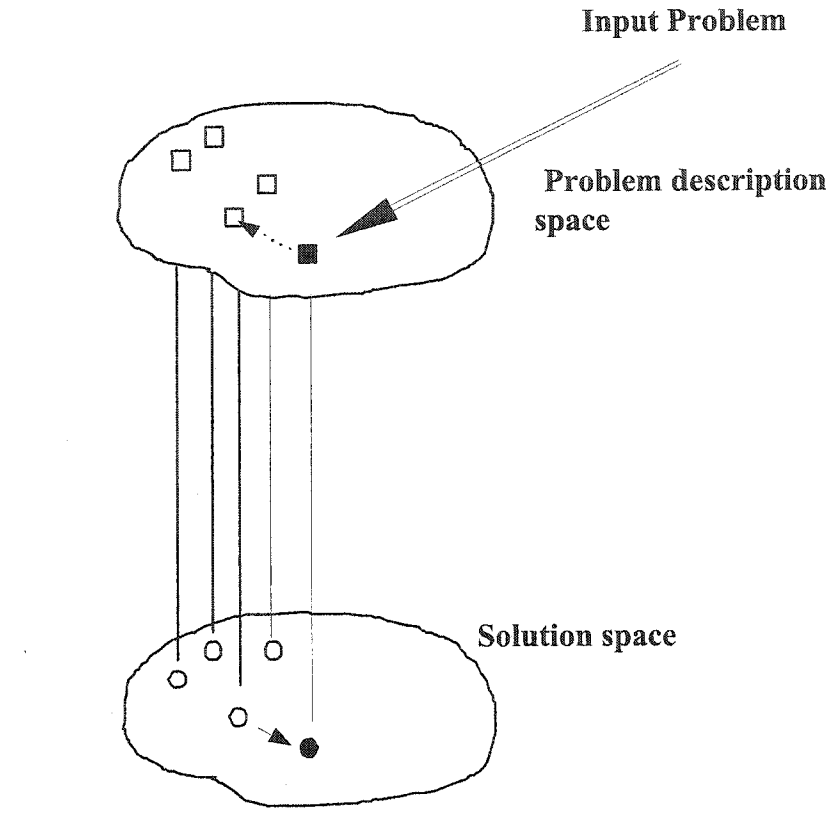
Learning is central to case-based planning because a case-based planner must reuse, and learn from, its own experiences to build new plans and to avoid past errors. For this purpose a planner remembers successes of plans so that they can be reused, it remembers failures of plans so that they can be avoided, and it also remembers repairs of plans so that they can be reapplied [16]. The new plan generation process is shown in fig. 2.2.

2.2.1 Main Steps in Case-Based Planning

Case-based planning can be viewed as being composed of the following steps:

- *Retrieval:* The planner searches the case memory for cases representing successful solutions for planning problems that are sufficiently similar to the current problem at hand. Typically, several candidates are found, and some ranking criteria are used to select one or more of them to be used in constructing the solution to the current problem. The amount of work performed in this phase depends on the philosophy of the planner, whether the strategy is anticipate-and-avoid or create-and-debug [16]: An anticipate-and-avoid type planner tries to ensure that the templates are as problem free as possible on the outset. A planner following the create-and-debug philosophy does not assign very strong quality criteria to the templates, but much work is done in the adaptation phase in order to obtain a quality plan.

- *Adaptation and Evaluation:*The template case(s) is (are) modified as to take the differences between the current planning problem and the one represented by the template into account. Adaptation techniques are manifold and depend highly on the problem domain. In addition to first principles planning methods, a few methods are characteristic to planning; those include merging partial solution from several template cases and interpolation between continuous valued plan parameters. Evaluation is a process that checks whether the adapted plan will, in fact, be suitable to solving the current case. This



■ Description of new plan to solve

□ Descriptions of plans solved previously

○ Stored solutions

● New solutions result ing from adaptation of stored solutions

Situation assessment maps from an input plan to a set of indices describing the plan

Retrieved searches for the case with the most similar plan description

Adaptation transforms the solution of the most similar plan into one that fits new circumstances

Figure 2.2: Process of a new solution (plan) generation in CBP

check can be interleaved with the adaptation and, given adaptation procedure that is complete, even be skipped altogether: a complete adaptation procedure always produces suitable plans.

- *Execution and Storage:* The planner tries out the plan in the world and records the outcome of the plan, i.e. what was the resulting world state like as compared to the goals. If the plan was successful, the planner has more latitude in deciding what to do: if the case was very similar to its template, it probably will not need to be stored in case memory. On the other hand, if the case required considerable amount of adaptation, then it probably will pay off to store the case. If the plan fails, the planner needs to do something: the planner understanding of the world model was faulty and needs to be updated somehow. Storing the case and indexing it by the failure is one approach for achieving that.

This division follows the "traditional" way of defining case-based planning. However, this view is perhaps too simple: case retrieval is not necessarily "unaware" of adaptation process. In fact, these two processes can be very much interleaved: the solution can be composed incrementally picking partial plans from case memory and merging them into the global plan by the means of adaptation [17].

2.2.2 The Need for Case-based Planning

In AI planning, general-purpose architectures have been developed. As with the expert systems, the main motivation of these architectures is to decouple knowledge representation from inference. In this way, the same architecture may be used, for example, to plan a sequence of actions to transport radioactive material between a nuclear plant and a soil or to transform a piece of raw material into a mechanical work piece.

One of the main difficulties with general-purpose architectures is generating adequate control strategies to plan for the sequence of actions solving a problem. The use of control rules is usually not a feasible choice for two reasons: first, whereas the knowledge about actions transforming the world can be modelled directly by observing processes in the problem domain, the knowledge about how to reason with such actions may be very difficult to acquire; extracting the latter knowledge results in the bottle-neck typical of knowledge engineering. Second, acquiring the control rules by learning from previous problem-solving episodes results in the utility problem [18].

Case-based planning (CBP) offers an alternative to rules for controlling the general-purpose architectures. In CBP, previously obtained solution plans and their problem descriptions are stored as problem-solution pairs; a similar case is retrieved and its solution is adapted to solve the new problem. Retrieval and adaptation are key aspects determining the performance of case-based planners cases.

2.2.3 Issues about Case-based Planning

In general, any case-based planner must address the following issues [19]:

1. *Similarity assessment:* The similarity assessment must predict if a given case can be easily adapted to solve the new problem.

2. *Organization of the case base:* The case base must be structured in a way that enables to evaluate the similarity assessment on the existing cases in an efficient way.

3. *Adaptation of cases:* Once one or more similar cases to the new problem have been retrieved, their solutions must be adapted to build a solution of the new problem.

Several general-purpose, case-based planners have been developed that assess one or more of these issues [16], [19], [20], [21], [22], [23] and [24]. Following are the main characteristics of case-based planners:

Goal Driven Retrieval: Planning problems are described as a pair (I, G) where I represents the initial conditions or features of the problem and G are the goals to be achieved [1]. Case based planners traditionally first examine the goals of the candidate cases, preselecting the ones that achieve one or more goals of the new problem. Comparing the initial conditions of the preselected cases makes the final selection. Most of the case libraries known in the literature reflect this principle by indexing the cases by their goals at the top level [19], [22], [24].

Static Similarity Metrics: For synthesis tasks such as planning, similarity metrics predict the adaptation effort of the cases to the new problem. Some similarity metrics take into account relevant features instead of all the features stated in the problem descriptions [19]. Others analyze the contribution of the features to a particular solution and rate them accordingly [20]. But common to all of them is the fact that the similarity metrics are static in that the measure between a case and a problem is always the same.

Relevance of Features: A case is usually seen as a pair $((I, G), Sol)$, where (I, G) is a problem description and Sol is a solution plan for (I, G). There can be several solution plans for the same problem. It has been observed that the relevance of a feature in I depends on the particular solution Sol [25]; whereas a feature may be relevant for a particular solution, it may not be relevant for another solution. A method, known as the footprinting process, has been developed to identify if a feature is relevant for a particular solution [19]. Base of this method is the goal regression process used in EBL [26].

Adaptation with Analogical Replay: Most of the adaptation methods in CBP are based on replay. Under this approach the cases are viewed as derivational paths indicating which decisions were taken at the choice points [25]. Initially, the method was implemented on a state space, case-based planner [19] but later it has been implemented in case-based, plan space planners as well [27] and [28]. During the adaptation phase the derivational path is reconstructed relative to the conditions of the new problem by taking the same decisions. This paradigm has been extended by

developing a language to perform annotations on the derivational path [19]. These annotations indicate failed decisions at the choice points. The language represents situations occurring during state space planning. For plan space planners only the replay method has been used but no attempts to express the failures have been made until now.

Tradeoff between Efficiency Gains and Case Search: Most case-based planners retrieve multiple cases to solve new problems [19], [20], [22] and [24]. Each case covers one or more goals of the new problem in pursue of covering as much of the goals as possible. Methods for merging them vary depending on the underlying planner. For state space planners, merging can be done by interleaving first principles planning and replay [19]. For plan space planners, replay is done first and then first principles planning follows [27]. This is based on the capability of plan space planners to decouple plan step execution from plan step ordering, which allows them to interleave steps in the plan. In general, however, the case-based planner should not spend arbitrary time searching for the cases to be retrieved; given that the search has time costs, there is a point where it doesn't payoff to further search for more cases [19].

Policy to Create New Cases based on Retrieval Failure: Most systems create new cases eagerly; every time a new solution is found, it is stored together with the corresponding problem description as a new case. Others follow a more elaborated policy: new cases are created only if the retrieved case is found not to fit into a solution plan of the new problem. That is, if parts of the case need to be

revised to find a solution plan. In such situations the retrieval is said to be a failure [24].

2.2.4 Case-Based Planner Architecture

The first architecture for case-based planner was proposed by Hammond [16]. This architecture is shown in figure 2.3.

A modified architecture for case-based planner was introduced in [29]. This architecture is shown in figure 2.4.

An alternative approach is to store and reuse traces of how those plans/solutions were derived, instead of the actual plans/solutions. By capturing and replaying the reasoning/planning trace involved in selecting planning operators rather than the planning steps themselves, the derivational analogy approach facilitates application of stored traces of processing to a wider class of problems.

The purpose of solving problems by analogy is to reuse past experience to guide generation of solutions for new problems avoiding a completely new search effort. Transformational analogy and most CBR systems reuse past solutions by modifying (tweaking) the retrieved final solutions as a function of the differences found between the source and the target problem. Instead, derivational analogy is a reconstructive method by which lines of reasoning are transferred and adapted to a new problem [25] as opposed to only the final solutions.

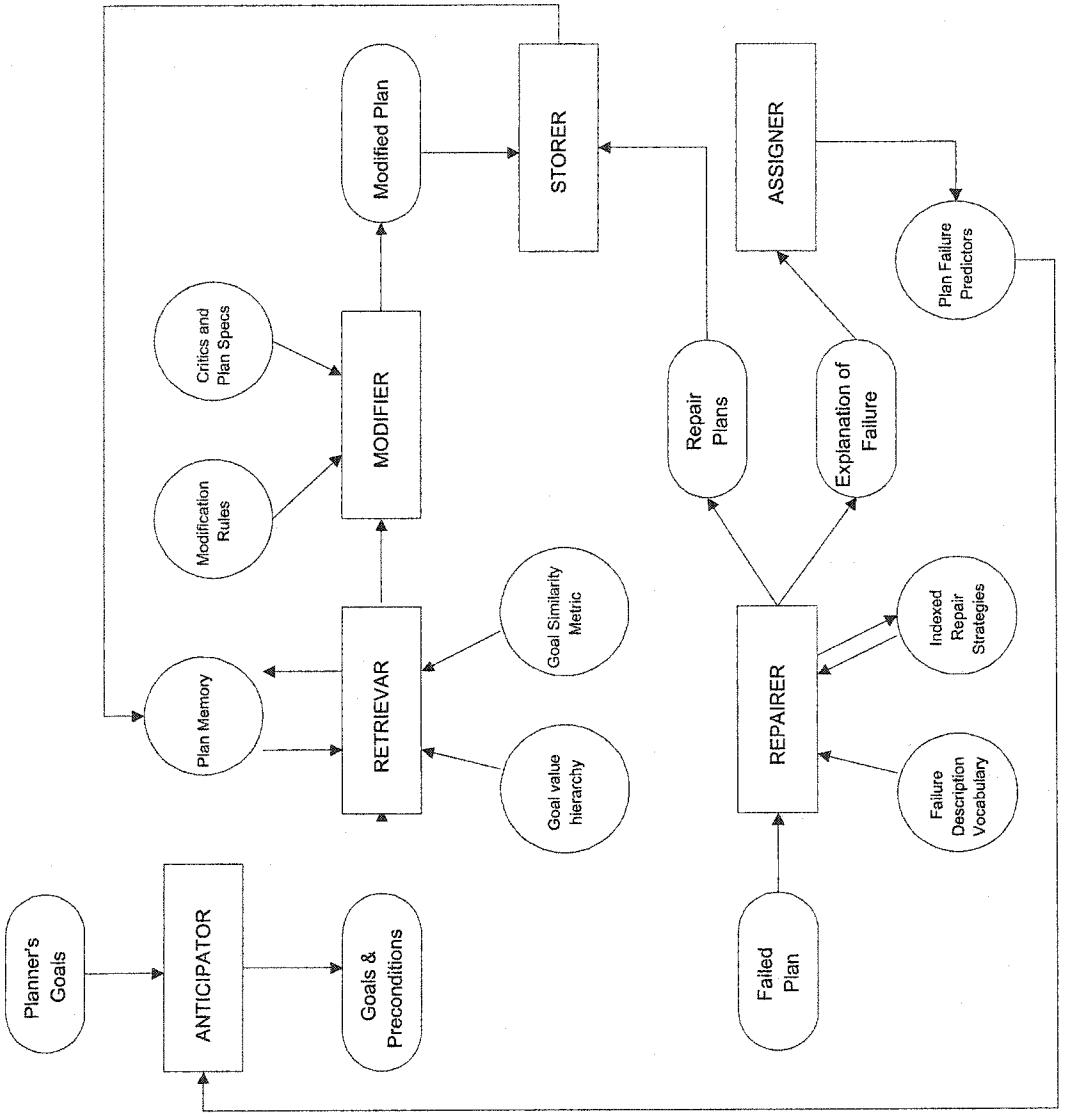


Figure 2.3: Case-Based Planner Architecture (Hammond, K., 1990).

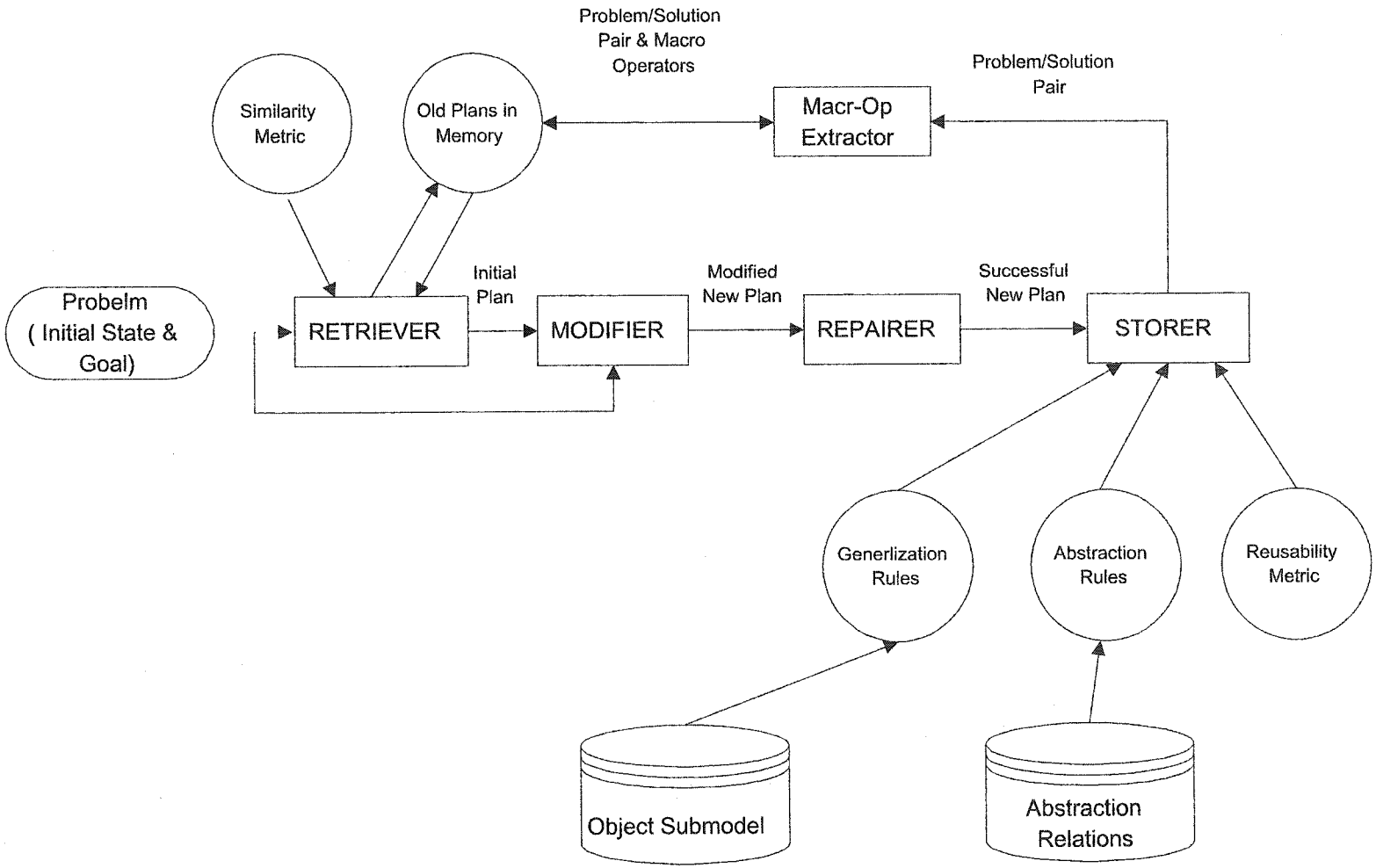


Figure 2.4: Case-Based Planner Architecture (Ahmed, M. and Rine, D. 1998)

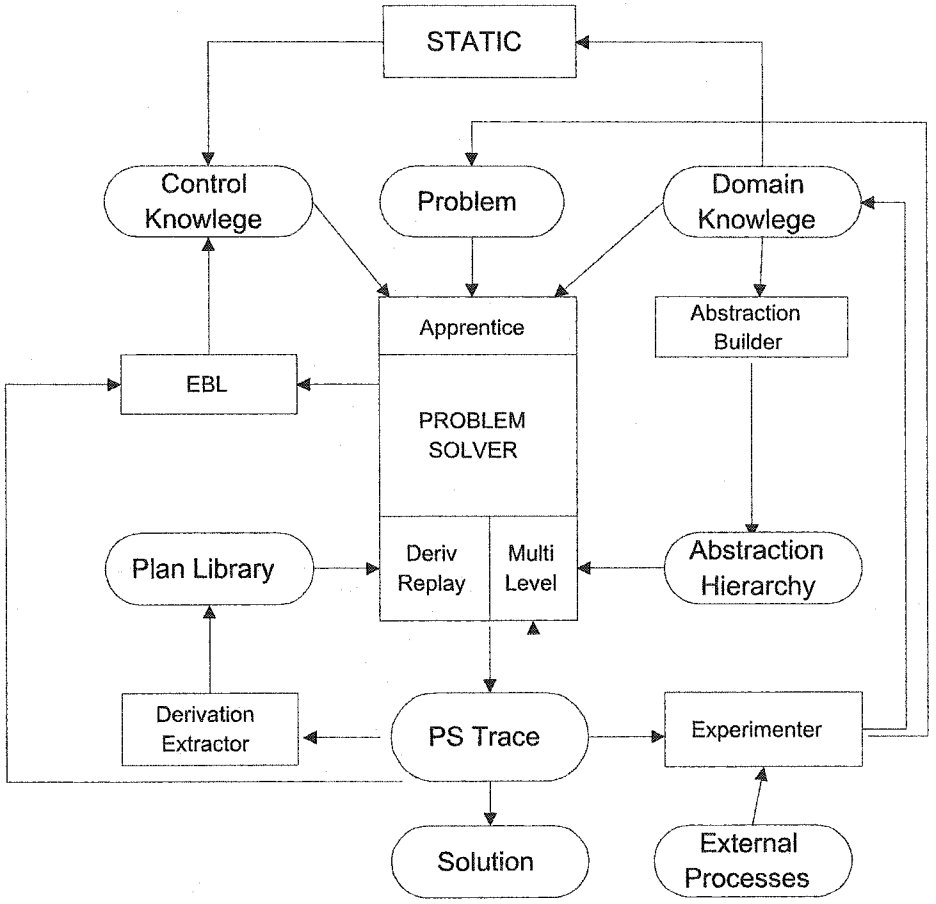


Figure 2.5: Prodigy/Analogy Architecture (Veloso, M. and Carbonell, J. (1993))

2.2.5 The Complexity of Case-based planning

Theoretical Analysis

The run-time complexity of case-based planning in STRIPS-type problems has been analyzed by [30]. They concentrate on the plan modification part as it usually dominates the complexity. They examine the decision problem versions of planning problems:

- PLANSAT: is there a plan that satisfies the goal.
- MODSAT: given a template plan P that solves a problem A, and a new planning problem B, decide if there is a way of transforming P into P' such that P' solves the planning problem B, and the size of the reused part is at least k.
- MODDEL: version of MODSAT such that insertions are not allowed in the middle of the template plan, i.e. new steps can only be added before and after the original plan.
- MODDELINS: version of MODSAT such that insertions are allowed in the middle of the original plan.
- MODMIX: the steps of the original plan need not to be in the same order as in the original plan.

They proved some interesting relations between worst-case complexities of the problems.

- MODSAT cannot be easier in the worst case than PLANSAT, even if the planning problem is restricted. The reason for this is that there is always the possibility that the template plan is empty. This result is not surprising and one can question its relevance. In particular, the authors note that there seems to be no general way of proving similar result for the situation where template is close to the current problem.

- In general, MODSAT is also not harder than PLANSAT, as both problems are PSPACE-complete or NP-complete, given the same restrictions of the planning problem. In other words, if the problem domain is difficult enough, the worst-case complexities of generative planning and plan reuse are the same. The same holds even if the template is guaranteed to be "one goal away" of the current problem- provided that the plan description is not allowed to be of exponential size.

- There are instances where PLANSAT problem can be solved in polynomial time but the corresponding MODDEL and MODELINS problem are NP-complete. In [30] this fact is acknowledged that the similarity of the template plan is not reflected in these formulations. However, they also were able to prove that even if we restrict ourselves to situation where the template and the current problem are one goal away, the result still holds.

In addition to these results, [30] show that the matching problem, i.e. finding the template case, can be hard, if we require finding the best template. **However,**

finding the provably optimal template is seldom required and one should note that these are worst-case results under classical planning assumptions and do not represent all practical situations. Particulary when the independence of goal increases, i.e. the problem domain gets simpler, the case-based approach gains over generative planning: finding a close template gets easier and so does adaptation of the template. In [31]and [32] adaptation guided retrieval of cases have been presented, which provides easiness in adaptation process and makes it less complex.

Empirical results

In the light of the preceding theoretical results, one could ask whether they have any bearing to performance in practical tasks. To the [30] knowledge empirical comparisons of case-based and generative planning approaches are few and far between.

- In [33], [34] as well as [30] the relative merits of generative planning and case-based planning in the blocks-world domain have been studied . In the PRIAR system of [33] plan reuse worked extremely well. As compared to the same system performing generative planning, timesaving ranged in between 34 and 99 percent. In [34] a similar experiment with SPA system is performed : their corresponding numbers were 18-67 percent; they give the poor generative planning ability of PRIAR as the reason for the differences. In [30] more mixed results are put forwarded : a case-based approach worked well when the modification tasks were simple; when in all cases all blocks were initially

clear and on table, only the number of them differed. Reusing plans was always faster than generating from scratch in these situations. When the modification task got more involved, i.e. some blocks were stuck under other blocks, generating the plan from scratch was sometimes up to several times more efficient.

- In [17], it is reported that running time for PRODIGY/ANALOGY is less than 15% of A* search and little under 90% of that of another heuristic planner when solving 30 route problems on the map of Pittsburgh. The quality of the plans (route length) generated by the case-based method was much close to the optimal (A*) quality than the quality obtained by the generative heuristic planner.

- [35] reports on results on case-based job-shop scheduling. Their CABINS system performed several times faster than a simulated annealing approach in improving an initial schedule. In addition, the quality of the CABINS generated schedules was better. One should note, however, that scheduling problems differ from other planning problems in that the goal in scheduling can be a complex objective function, and thus hard to obtain exactly; for example the complex simulated annealing algorithm rarely finds the optima.

Based on these experimental results, it seems that case-based methods can improve also the run-time efficiency. A number of case-based planners exist, as discussed in next section, which provide promising results and gain in efficiency as compared to

planning from scratch.

2.2.6 Existing Case-Based Planners

Several general purpose case-based planners have been developed including:

CHEF: CHEF is the first case based planner developed by Hammond [36]. Its approach to planning is to make use of memory whenever possible. It begins planning by using its memory of past failures to warn problems. It then uses memory of success to modify them (if needed) for its current goals. In case of plan failure it explains the failure otherwise it stores the successful plan in memory.

Prodigy/Analogy: A pioneer work in the field is [25] and [19]; it implements for the first time a complete general purpose case-based problem solving cycle. Novel features include adaptation with analogical replay, the concept of relevant features and a complete architecture of the case base. Prodigy/Analogy is based on the state space planner Prodigy [37].

Priar: Another early work in the field [20]; some of the novel features in Priar are cases representing a hierarchical plan, adaptation of hierarchical plans and rating of the features by their contribution to the cases.

SPA/MPA: SPA is a case-based planner performing single case adaptation [34] and MPA is the extension of SPA performing multiple case adaptation (Francis and Ram, 1995b). The most important contribution of this work is an adaptation algorithm

in which the solution of the case is transformed into a solution of the new problem. This is opposed to adaptation based on replay in which the derivational trace is reconstructed relative to the new solution. Unfortunately, even though SPA has been shown to outperform first principles planning with SNLP, no report has been made comparing it with adaptation based on replay.

derSNLP+EBL: The case-based planner derSNLP+EBL performs case adaptation with standard replay based on SNLP [24]. If a retrieval failure occurs, EBL generates a rule explaining the failure. Each of the retrieved cases is annotated with the rule. The censoring rule serves as a choice node in the case base between each of them and a new case containing the solution obtained when the cases were retrieved and the failure occurred. In subsequent retrieval episodes, before selecting any of the censored cases, the retrieval procedure checks that the rule does not hold. Otherwise, the new case is retrieved. The new case may be censored as well, if it is retrieved and a retrieval failure occurs.

MRL: The inference mechanism of MRL is based on deductive planning [21]. The deduction mechanism is used to generate plans. The functionality obtained is somehow comparable to state space as the state of the world is represented and transformed in the deductive formalism.

Paris: Paris reuses abstract cases [23]. Given a new problem, its problem description is abstracted. The case base is searched for a case solving the abstracted problem

description. Once such a case is found its solution is refined to a concrete level. Abstracting the solutions found creates new cases. Paris is based on a linear, state space planner.

CAPLAN/CBC: In CAPLAN/CBC [18] retrieval is twofold integrating static and dynamic retrieval techniques. The adaptation method is complete decision replay, an extension of standard replay for plan-space planners. New cases are created only if during case-based planning episodes the guidance provided by the available cases is considered nonbeneficial.

Conversational Case-Based Planner In [38] a tool for case-based planning is discussed that support user interaction during case retrieval and adaptation process. Also in [39] a planning system *SiN* has been described which is an integration of conversational case retrieval with generative planning.

2.2.7 Retrieval Step in CBP

Retrieval is the first and most important process in case-based planning and case-based reasoning. Case-based planning begins with cases and cases are obtained by retrieval. The basic problem in retrieval is to find similar cases to find good analogs. We focus on how to improve the process of finding good analogs in because the obtainment of good analogs will reduce the need for adaptation.

Given a description of a plan, a retrieval algorithm should retrieve the most similar plans to the current problem or situation. The retrieval algorithm relies on the

indices and the organization of the memory to direct the search to potentially useful cases. The issue of choosing the best matching case has been addressed by research into analogy [40]. This approach involves using heuristics to constrain and direct the search. Several algorithms have been implemented to retrieve appropriate cases, for example: serial search [41], [42], [43], hierarchical search [44] and simulated parallel search [45]. Unlike database searches that target a specific value in a record, retrieval of cases from the case-base must be equipped with heuristics that perform partial matches, since in general there is no existing case that exactly matches the new case.

2.3 Planning with imperfection

2.3.1 Probability-based planners

There are a number of planners based on probability like:

Buridan: Buridan [46] is a modified version of the SNLP [47] planning algorithm that can create plans that meet a threshold probability of successes when actions are non-deterministic. Buridan differs from SNLP by allowing more than one causal link for each condition in the plan.

DRIPS: In DRIPS [48] ranges of utility values are computed for partial plans, encompassing the best and worst expected utilities of all possible completions of the partial plans. DRIPS uses skeletal refinement planning base on an abstraction hierarchy of operators. In this approach, a partial plan is a sequence of operators, one or more of which may be an abstraction of a number of ground operators.

Planning proceeds by repeatedly choosing a more specific version of an operator in the plan until there are no abstract operators left.

Weaver: Weaver [49] is a probabilistic planner based on Prodigy [19]. It has no representation of utilities, and has no explicit model of observability, but assumes the world is completely observable at plan execution time. It has an explicit representation for uncertain exogenous events.

MAXPLAN [50] and *Cypress* [51] are some other examples of probabilistic planners.

2.3.2 Fuzzy logic-based planners

Fuzzy logic has been used for dealing with imperfect information in many planning and control systems. A planning architecture for intelligent robot, based on fuzzy memory-based reasoning for real time planning/control is proposed in [52]. This architecture is based on the idea of memory-based reasoning systems and behavior-based control systems. It consists of two different planners, namely Reactive planner (which selects and executes the best control strategies which are beforehand prepared) and Macro planner (which makes rough and robust planning by matching between the trigger and goal conditions of control agents).

In [53] a modified form of the above system is presented for handling fuzziness. The planner of this system constructs a tree-structured plan by a best-first algorithm giving priority to reliable path, and memorizes it. Based on the sensory information about the users, the controller executes the plan, selecting the most reliable and

useful path of the tree.

In [29] a fuzzy logic-based framework for designing control for totally autonomous vehicles is presented. This framework is based on cooperative intelligent agents that cooperate through a blackboard. It has also introduced an approach for designing on-line fuzzy-logic controllers using planning techniques. This design approach is based on plans reuse.

Autonomous vehicle motion planning [54], and Path planning and execution [55] also use fuzzy logic.

2.4 Fuzzy Logic and Case-based Systems

In case-based systems, the use of Fuzzy Logic techniques may be relevant, in case representation to allow for imprecise and uncertain values in features, and case retrieval by means of fuzzy matching techniques [56]. Moreover perhaps the most severe limitation of existing systems [all types of Case-Based systems] is the feature-value representation that is being used for cases. This representation is in the form of crisp value. The consequence is that case-based algorithms cannot be applied to knowledge-rich applications and imprecise environment that require much more complex and imprecise case representations. The retrieval process in case-based systems can be carried out by the fuzzy logic process of *measuring the degree of similarity of cases.*

There are a number of case-based reasoning (CBR) systems, which use fuzzy logic

in different respects e.g. case representation, for matching cases etc. Some of the existing fuzzy-based CBRs are discussed below.

ARC System: In the ARC system [57], memory organization is based on a hierarchy of categories and cases. A category is formed by a fuzzy prototypical description of features common to most cases belonging to the category. Each category is linked to its subcategories, a set of differentially indexed cases, and near-misses cases. The retrieval process, first, selects the most promising categories by a pattern-matching algorithm [58] based on common features between the problem and the categories. Then cases are selected using differences between the categories and themselves.

FLORAN System: In the FLORAN system [59], cases and problems are represented in an object-oriented environment. Therefore cases and problems are instances of classes. Classes define slots that have a fuzzy value. In the selection step, the current problem is compared with each filtered case using a fuzzy pattern-matching algorithm [58].

CAREFUL System: CAREFUL system [60] uses possibility theory for retrieval step in CBR. For an efficient retrieval, a hierarchical fuzzy classification algorithm is adapted for case filtering. Weighted fuzzy pattern matching algorithm [61] is used for selection step.

Weather prediction CBR system [62]: This system uses fuzzy k-nearest neighbor algorithm [63] for the selection (retrieval) step.

In [64], the design issues for fuzzy case-based reasoning system are discussed.

2.5 Conclusion

In this chapter, we have presented different representation approaches for imperfection in knowledge. Then we presented one of the knowledge-based planning technique, namely case-based planning. Different issues related to case-based planning were also discussed. Finally we discussed planning with imperfection and case-based systems with imperfection in particular.

Chapter 3

Similarity Measuring Techniques in Case-based Systems

The similarity information in case-based planning can be divided into two kinds. One is called qualitative similarity information which represents the similarities between cases i.e. for a chosen case pair (x,y) if case x is similar to case y or not. The other is called relative similarity information which represents the relation between similarities of two case pairs both including the common case i.e. for each chosen set of three cases (x,y,z), if x is more similar to y than to z.

There are different issues in similarity assessment of cases in CBP; one is how to determine the right features to compare. Decisions about which features are important are often based on explanations of feature relevance, but those explanations may be imperfect, leading to a need for robust similarity metrics that take the difficulties in specifying important features into account Another problem is that for

some tasks, input problem descriptions are not sufficient to determine the similarity of old and new situations.

A single set of static similarity criteria may not capture the right distinctions among cases. Different cases may be most appropriate to consider depending on the relative importance of different dimensions for judging the success of the CBP process. For example some of the criteria might be reliability of the resulting plan, execution time for that plan, the time required to generate the solution, or even, if a creative solution id desired, the novelty of the result.

While some case-based approaches retrieve a previous case largely based on superficial, *syntactical similarities* among problem descriptors, some approaches attempt to retrieve cases based on features that have deeper, *semantical similarities.* Syntactic similarity assessment (knowledge poor approach) has its advantage in domains where general domain knowledge is very difficult or impossible to acquire. Semantical oriented approaches (knowledge intensive approaches) are able to use the contextual meaning of a problem description in its matching, for domains where general domain knowledge is available.

- **Surface Similarity**:According to [65], surface similarity is based on readily accessible components of objects. Gentner [66], treats surface similarity in object attributes as opposed to the similarity in relations. In [67] surface similarity is called as *perceptual similarity* and is defined as cognitively primitive and well defined. Vosniadou [68] suggests using *salient similarity* instead of

surface similarity to refer to similarity grounded in easily retrievable aspects of representations. It does not need to be descriptive attributes. According to her, the surface/deep similarity distinction is a dynamic one because over time the representations may change and thus may change salience of attributes.

- **Deep Similarity**: According to [65], deep similarity is based on more central, core properties of objects. Ross [69] observed that even though surface similarity is used more often, structural similarities are more successful in solving the new problems. Gentner [66] considers deep similarity as structural similarity, similarity based on the relational structure. Vosniadou [68] notices that relational properties of objects may be particularly salient even for young children. This shows not only that surface similarity can be salient, but that relational properties can also be easily accessible.

It was observed that experts are able to perceive the underlying similarities in problems, whereas novices are not capable to see same similarities mainly because the difference in surface attributes [68].

As for the connection between surface and deep similarities, it is claimed in [65] that the surface similarity serves as a heuristic for where to look for deeper attributes. Another use of surface similarity for deep attributes is to constraint on the predicates that compose the mental representations.

3.1 Similarity Measuring Techniques

3.1.1 Context-based similarity

In [70] a context-based similarity is applied for retrieval of relevant cases. Three kinds of dependencies of similarity assessment on context are mentioned: permanent independent similarity ,temporally independent similarity and context dependent similarity. For retrieval only those items are considered that are similar to the query with respect to current context. Context is defined as

A finite set of attributes with associated constraints on the attribute values. The constraints on attribute values are specified either as a "allowed" values (e.g. values which should be present if the attribute matching is to occur) or as a "prohibited" values (e.g. values considered which should not be present if the attribute matching to occur).

Similarity is defined as a relation of three parameters: *set of items* (SI), *a context* (Ω) *and an information base* $(\Delta \supseteq SI)$.

3.1.2 Nearest Neighbor

This approach involves the assessment of similarity between stored cased and the new input case, based on matching a weighted sum of features. The biggest problem here is to determine the weights of the features. The limitation of this approach includes problems in converging on the correct solution and retrieval times. In general the use of this method leads to the retrieval time increasing linearly with

the number of cases. Therefore this approach is more effective when the case base is relatively small. A typical algorithm for nearest neighbor is used by Cognitive Systems ReMind software reported in [71], where w is the importance weighting of a feature, *sim* is the similarity function, and *fI* and *fR* are the values for feature i in the input and retrieved cases respectively.

$$\frac{\sum_{i=1}^{n} w_i \times sim(f_i^I, f_i^R)}{\sum_{i=1}^{n} w_i}$$

3.1.3 Template-Based Retrieval

Similar to SQL-like queries, template retrieval returns all cases that fit within certain parameters. Takehito Utsuro [72] has also proposed the concept of similarity template for optimizing nearest neighbor algorithm for retrieval. This technique is often used before other techniques, such as nearest neighbor, to limit the search space to a relevant section of the search space to a relevant section of the case-base. Similarity function for similarity template is defined as

$t = \langle s_1, ..., s_n \rangle$

$$sim_t(t) = \frac{\sum_{i=1}^{n} w_i \times s_i}{\sum_{i=1}^{n} w_i}$$

where s_i denotes the similarity of nearest neighbor formula. In the method, generating retrieval queries from an input and similarity templates in a certain order optimizes the nearest neighbor retrieval process.

3.1.4 Induction

Induction algorithms [73] determine which features do the best job in discriminating cases, and generate a decision tree type structure to organize the cases in memory. This approach is useful when a single case feature is required as a solution, and where that case feature is dependent upon others.

3.1.5 Footprint based Similarity

In [25], two similarity metrics are defined: Direct Similarity Metric (DSM) and Footprint Similarity Metric (FSM). In DSM each past problem-solution pair is indexed by the corresponding initial predicates and goal statement predicates. When a new problem is given to the system in terms of its initial state predicates and goal statement predicates, the system look for the past problem-solution pair that has the same predicates of the new problem. Once the system found past problem-solution pair that has the same predicates, it does a permutation to map the new-problem-initial-state and the new-problem-goal-statement parameters with the old-problem-initial-state and the old-problem-goal-statement. The combination that gives the maximum percentage of the predicates that match together is used. This percentage represents the degree of similarity between the past problem and the new problem.

The Footprint Similarity Metric captures into the similarity metric the role of the initial state in achieving the different goals conjuncts with respect to a particular solution found. The FSM is similar to DSM, the only difference is that it measures

the degree of similarity between the predicates of the initial state that have been used to achieve the solution but not all the initial state predicates. Predicates of the initial state that have been used to achieve the solution are called "the footprints". Interacting foot-printed similarity metric for PRODIGY/ANALOGY is defined by [19] as:

Let P be a new problem and P' be a previously solved problem, respectively with initial states S^P and $S^{P'}$, and goals G^P and $G^{P'}$. Let δ_G^σ be the match value (number of matched literals) of G^P and $G^{P'}$, under substitution σ ,such that the matched goals $G_1, ..., G_m$ cover completely one or more sets of interacting goals. Let $S_{fp}^{P'}$ be the foot-printed initial state of problem P' for the set of matched goals $G_1, ..., G_m$. Let δ_S^σ be the number of literals of $S_{fp}^{P'}$ matched with literals of S^P under substitution σ.

The two problems P and P' interactively foot-print match with match value $\delta_\sigma = \delta_G^\sigma + \delta_S^\sigma$ for substitution σ.

The same footprint similarity metric is also used in DERSNLP (derivation replay in SNLP) [24].

3.1.6 Explanation-based similarity

In PARIS [74] explanation-based similarity has been used for retrieval of similar cases. To enable explanation-based similarity assessment the single level explanations of the case in the case base are attempted to be mapped to the current problem description. The procedure of similarity assessment between a complete case and a

problem starts by the attempt to map explanations at the highest level of abstraction. If the mapping is successful, the process proceeds with the next, more concrete level. The lowest level at which the explanations can still be mapped, indicates the degree of similarity between the case and the current problem.

3.1.7 Weighted foot-printed similarity

In CAPLAN/CBC [18] a dynamic similarity metric is proposed in conjunction with static retrieval (dependency-driven retrieval) technique. Dependency driven retrieval is the technique developed to handle problems that are given in the form of extended problem descriptions $(I, G, \prec)$. To take advantage of this information, the dependencies, $<_C$, between the goals achieved in each case C are represented explicitly. The dependencies reflect the order in which the goals are achieved in C and they are represented at the top level of the case base in CAPLAN/CBC. During retrieval, the dependencies $<_C$ between the goals in the cases are compared against the ordering restrictions $\prec$ of the problems. This means that retrieval in CAPLAN/CBC is driven by the dependencies instead of the goals as in other case-based planners.

For Dynamic Similarity Metric in CAPLAN/CBC features are given a weight. These feature weights represent a hypothesis about the relative relevance of the feature in the case. The similarity metric counts these feature weights during retrieval to determine if the case should be retrieved to solve the current problem. Once a case is retrieved, CAPLAN/CBC evaluates if the retrieval was adequate. If this is the situation, the hypothesis about the relevance of the features is reinforced. Oth-

erwise, the hypothesis is punished. Updating its weight makes reinforcement and punishment of a hypothesis about the relevance of a feature. That is, the feature weight is increased or decreased relative to the other feature weights in the case. The updated weights indicate again a new hypothesis about the relevance of the features in the case and this hypothesis is tested in future retrieval episodes.

The weighted similarity metric between a case C and a problem P,$sim^{wg}(C,P)$, is defined as:

$$sim^{wg}(C,P) = \begin{cases} \sum_{i \in I \bigcap_\theta IC} w_{i,c} & G_C\theta \subset G \\ 0 & \text{otherwise} \end{cases}$$

where $I \bigcap_\theta IC$ denotes the set of all features in I_C matching a feature in I with a substitution θ . The weight, $w_{i,C}$, of a feature $i$is updated according to following equations,

$$w_{i,C} = \begin{cases} w_{i,C} + \Delta_{k^C,f^C} & \text{failed retrieval} \\ w_{i,C} - \Delta_{k^C,f^C} & \text{adequate retrieval} \end{cases}$$

Context of features has been determined by the domain theory in CAPLAN/CBC.

3.1.8 Fuzzy k-nearest neighbors technique

A fuzzy k-nearest neighbors (fuzzy $k-nn$) technique [63] is simply a nearest neighbor technique in which the basic measurement technique is fuzzy. The fuzzy $k-nn$

technique applied to continuous vectors, achieves automatic, expert-like similarity comparison of complex objects. Each comparable attribute in two comparable vectors is compared with an attribute-specific fuzzy set and the results are aggregated to describe the overall similarity of the two vectors.

Stored cases are labeled into distinct classes in fuzzy k-nearest neighbors algorithm. Given a new case to classify, both crisp and fuzzy $k-nn$ algorithms are made to find $k-nn$. The algorithms differ in two ways. First, the crisp algorithm uses a distance function, whereas the fuzzy algorithm uses fuzzy set based comparisons. Second, the crisp algorithm assigns the new case to the class that is represented by a majority of its $k-nn$, whereas the fuzzy algorithm assigns the new case varying degrees of membership to all the classes represented by the $k-nn$, according to the degree to which the new case matches each of the $k-nn$.

In [63] the performance of the fuzzy k-nn nearest neighbors with crisp $k-nn$ algorithm (k-means clustering) are compared and found two advantages in fuzzy $k-nn$ method:

- Fuzzy $k-nn$ classification is more accurate than crisp $k-nn$ classification.
- Fuzzy $k-nn$ classification solutions include useful confidence measures based on to the resulting memberships.

3.1.9 Fuzzy predicate-based similarity metric

In [29] a fuzzy-based similarity metric has been proposed for case-based planning in imprecise environment. The pre-conditions and post-conditions of a plan are represented by fuzzy predicates. Also an importance value (IV) is assigned to each predicate showing its importance for overall plan execution.

$$SM_{ji} = max_{PR} \frac{\sum_i IV_i.PSM_{ji}}{\sum_i IV_i}$$

where

$$PSM_{ji} = \begin{cases} x + \mu_i(1-x), & \forall P_i \in (OR_j \cap NI) \\ 0 & \forall P_i \in (OR_j - NI) \end{cases}$$

$$PSM_{ji} = \begin{cases} x + \mu_i(1-x), & \forall P_i \in (OS_j \cap NS) \\ 0 & \forall P_i \in (OS_j - NS) \cup (NS - OS_j) \end{cases}$$

However it does not exactly defined the updating mechanism for IV. Also experimental results have not been conducted to verify this SM. Moreover weights are assigned to all the predicates.

3.1.10 Measure base on the operations of union and intersection

In [75] the grade of similarity $M_{A,B}$ of the fuzzy sets A and B, is defined by

$$M_{A,B} = \frac{\sum_i (a_i \wedge b_i)}{\sum_i (a_i \vee b_i)}$$

where $\wedge$ and $\vee$ are the fuzzy intersection and union operators respectively.

3.1.11 Fuzzy Integration for similarity metrics

In [76] a general method of similarity metrics is proposed by focusing the similarity of features of problems that are represented by frame knowledge expressions. In this method a degree of similarity is assigned as a retrieving criterion between the features of new problem and past one, and using fuzzy integration to calculate the degree of similarity by comparing target frame that presents the features of past problem. A comparative frame is also made which includes the information of matching source frame to target frame for adaptation. Degree of similarity of simple frame f_t and f_s is denoted by $DS_{f_t \sim f_s}$ defined as follows:

$$DS_{f_t \sim f_s} = \int f_t \sim f_s \circ g$$

A number of similarity-based fuzzy reasoning methods have been proposed in past. Many existing fuzzy reasoning methods are based on Zadeh's Compositional Rule of Inference (CRI), which requires setting up a fuzzy relation between the antecedent

and the consequent part. There are some other fuzzy reasoning methods that do not use Zadeh's CRI.

3.1.12 Chen's Matching Function (MF) Method

In [77] the degree of similarity is defined as

$$S_{MF}(A, A') = \frac{|A||A'| \cos\theta}{max(|A||A|, |A'||A'|)}$$

where $|A|$ and $|A'|$ are the length of the vectors A and A' , respectively, and $\cos\theta$ is the cosine of the angle between the vectors. The vectors A and A' represent the values of the fuzzy quantifiers in the symptoms of the diagnostic rule and the values of the fuzzy quantifiers provided by the user, i.e. the given facts, respectively.

3.1.13 Chen's Function T (FT) Method

In [78], Chen has proposed another fuzzy reasoning method based on the function $T(a_i, a_i') = 1 - |a_i - a_i'|$. Furthermore, weights have been included for each proposition in the antecedent of fuzzy production rules. The similarity between two vectors A and A', representing the values of the fuzzy quantifiers in the symptoms of the diagnostic rule and the values of the fuzzy quantifiers provided by the user respectively, is now measured by the similarity function $S_{FT}(A, A', W) \in [0, 1]$ defined as

$$S_{FT}(A, A', W) = \sum_{i=1}^{n} \left[T(a_i, a'_i) * \frac{w_i}{\sum_{k=1}^{n} w_k} \right]$$

3.1.14 Yeung et al.'s Degree of Subsethood (DS) Method

In [79], a fuzzy reasoning method has been proposed that employs similarity measure based on the degree of subsethood between the propositions in the antecedent and the given facts.

Degree of subsethood is defined as

$$S_{DS}(a'_i, a_i) = \frac{M(a'_i \bigcap a_i)}{M(a'_i)}$$

and the weighted average degree of subsethood function is defined as

$$S_W = \sum_{i=1}^{n} S_{W_i} = \sum_{i=1}^{n} \left[S_{DS}(a_i, a'_i) * \frac{w_i}{\sum_{j=1}^{n} w_j} \right]$$

Other methods can be found in [80].

3.2 Conclusion

In this chapter, literature related to similarity measuring techniques for case-based systems has been reviewed. These similarity measuring techniques include both fuzzy logic based and non-fuzzy logic based approaches. The similarity metrics based on fuzzy logic accept a crisp real value input between an interval of 0 and 1.

Only the metric proposed in [29] accepts a fuzzy value (set) as input. Our problem is to develop a similarity metric for retrieving effective similar cases for such domains where information is not precisely specified.

Chapter 4

Solution Approach

4.1 Introduction

This chapter mainly discusses our solution approach for developing a fuzzy predicate based dynamic similarity metric (FDSM) and the enhancement to the CBP architecture described by Ahmed, M. et. al. in [29]. Also we'll explain weighting model used by FDSM for adjusting the weights of case initial state predicates. In addition, details of the algorithms used by 'modifier', for mapping fuzzy predicates of a case to a problem, will be given.

In case-based planning (CBP), a new problem is solved by adapting one or more similar case(s) from a case base. The case-based planner searches the case base for finding "sufficiently" similar cases to the current problem by using some kind of similarity metric, which plays an important role for the whole case-based planning process. If similarity value of a case is greater than a certain threshold, which shows

that the case is sufficiently similar to the problem, then the case is retrieved from the case base. The solution(s) of the similar case(s) are then adapted/repaired to solve the new problem. For a fuzzy case-based planning system, retrieval process requires a similarity metric that could capture the imprecise information for matching.

In chapter 3 we have discussed a number of similarity metrics used in CBP systems, with their limitations. All of those are based on crisp logic and most of them are static. Although there are some similarity-based fuzzy reasoning methods available yet no one is specific to case-based planning. For finding the similar cases in fuzzy case-based planning systems, we have developed a dynamic similarity metric based on fuzzy predicates, discussed in section 4.2 . This metric is a modified version of the similarity metric presented by Ahmed, M. et.al. in [29]. We have also modified the CBP architecture of Ahmed, M. et. al. presented in [29] for similarity assessment by providing a feedback from repairer, which is explained in section 4.3. Section 4.4 describes the weighting model used for adjusting the weights of case initial state predicates which is followed by the details of the Tabbu Search (TS) [81] and Simulated Annealing (SA) algorithms used by modifier for mapping case/problem predicates in sections 4.5 and 4.5.2 respectively. Section 4.6 gives an illustrative example for mapping case/problem predicates.

4.2 Fuzzy Predicate-Based Dynamic Similarity Metric

Our fuzzy predicate-based dynamic similarity metric (FDSM) between a new problem a and a case (old Problem-Solution pair) b can be calculated as follows:

$$FDSM_{ab} = W_g.max_{PR}\frac{\sum_{j=1}^{n_{g_a}} wt_{j_{g_a}}.PSM_{j_{g_{ab}}}}{\sum_{j=1}^{n_{g_a}} wt_{j_{g_a}}} + W_p.max_{PR}\frac{\sum_{i=1}^{n_{p_b}} wt_{i_{p_b}}.PSM_{i_{p_{ab}}}}{\sum_{i=1}^{n_{p_b}} wt_{i_{p_b}}}$$

where

$PSM_{g_{ab}}$: goal predicates similarity measure of a and b;

$PSM_{p_{ab}}$: initial state predicates similarity measure of a and b;;

wt_g : weight for goal predicates of a new problem a;

wt_p : weight for initial state predicates of an old problem-solution pair b;

W_g : weight factor for goals;

W_p : weight factor for initial state; $W_g + W_p = 1$.

4.2.1 Initial state and goal predicates similarity measure

Let FP be the set of footprint predicates that appear in the initial state (precondition) of an old problem-solution pair. Footprint predicates are those initial state predicates of a case, which are used to achieve the solution. Generally these do not include all initial state predicates. We are using footprint predicates for

matching instead of all initial state predicates [29]. This is because the use of footprint predicates reduces the search space and time for matching the initial state of a case with initial state of a new problem [25]. Let OS be the set of all predicates that appear in the post-conditions (goal) of an old problem-solution pair. Let NI be the set of all predicates that appear in the initial state of the new problem. Let NS be the set of all predicates that appear in the goal statement of a new problem.

In order to use developed similarity metric, each past problem-solution pair is indexed by the corresponding footprint predicates and goal predicates. When a new problem is presented to the system in terms of its initial state's predicates and goal statement predicates, the system looks for old-problem-solution pairs that have high similarity with the new problem. Ideally

$$FP \subseteq NI;$$

$$NS = OS.$$

In a case base, for each old problem-solution pair PS_b, initial state predicates similarity measure $PSM_{p_{ab}}$ and goal predicates similarity measure $PSM_{g_{ab}}$ are calculated as follows [29].

$$PSM_{p_{ab}} = \begin{cases} x + \mu(1-x), & \forall P_i \in (FP_b \cap NI) \\ 0 & \forall P_i \in (FP_b - NI) \end{cases}$$

$$PSM_{g_{ab}} = \begin{cases} x + \mu(1-x), & \forall P_j \in (OS_b \cap NS) \\ 0 & \forall P_j \in (OS_b - NS) \cup (NS - OS_b) \end{cases}$$

In the above equations μ is calculated by taking the fuzzy intersection between a fuzzy predicate of an old problem-solution pair and one of a new problem. It is the maximum degree of membership that belongs to the fuzzy set that is the result of the fuzzy intersection. For $PSM_{p_{ab}}$ and $PSM_{g_{ab}}$, $x \in [0, 1]$ is a constant which can be adjusted according to the performance of similarity metric. x is used to differentiate between the case in which $P_i \in ((FP_b \cap NI) \cup (OS_b \cap NS))$ and $\mu = 0$ and the case in which $P_i \in ((FP_b - NI) \cup (OS_b - NS) \cup (NS - OS_b)$, where P_i represents a fuzzy predicate. Note that, before performing the fuzzy intersection between two fuzzy predicates they must be instantiated appropriately.

4.2.2 Weight for goal and initial state predicates

In our similarity metric, weight is assigned to each initial state predicate of an old problem-solution pair, denoted by wt_p and to the goal predicates of a new problem, denoted by wt_g.

Let's first discuss the weight for initial state predicates of an old problem-solution pair i.e wt_p. Informally, a predicate is considered relevant to a goal with respect to a solution, if the predicate contributes to achieve the goal in the solution [19]. However, not all the relevant predicates have same importance in achieving the goal.

Some predicates are more important for a solution than the others. For example, a predicate can contribute to execution of more than one action. wt_p expresses how much an initial state predicate is important for overall execution of the plan. This predicate weight represents the relative importance of the predicate in the case. Initially, when no information is given about the predicates' contribution in achieving the solution, all predicates are assumed to have same importance in a case. Next, by statically examining the predicates' contribution in achieving the solution, the weights of these predicates are updated. One way to do this is as follows; the weight of a predicate is updated by counting the actions N_As for which it occurred as a precondition while taking those actions dependent on N_As into consideration, at the same time. Note that the direct and indirect actions should be differentiated, so that more weight should be given to direct dependent actions. Later, predicates' weights are modified by a feedback mechanism that provides some information about predicates' importance during plan execution. In our CBP architecture (discussed in section 4.3), repairer provides feedback to similarity assessment component for updating the predicate weights. The value for wt_p is a real number between $[0, 1]$, initialized with 1 for all initial state predicates.

For example, in the battle-field domain (see section 5.3.3) , to destroy an enemy unit $U1$ near a location $L2$ using a fighter plane $P1$, the required sequence of actions is shown in figure 4.1.

In the plan of figure 4.1, pre-conditions for firing action at a location on a unit are mentioned in a dashed box. Among these pre-conditions, $PHaveG(P1, G1)$

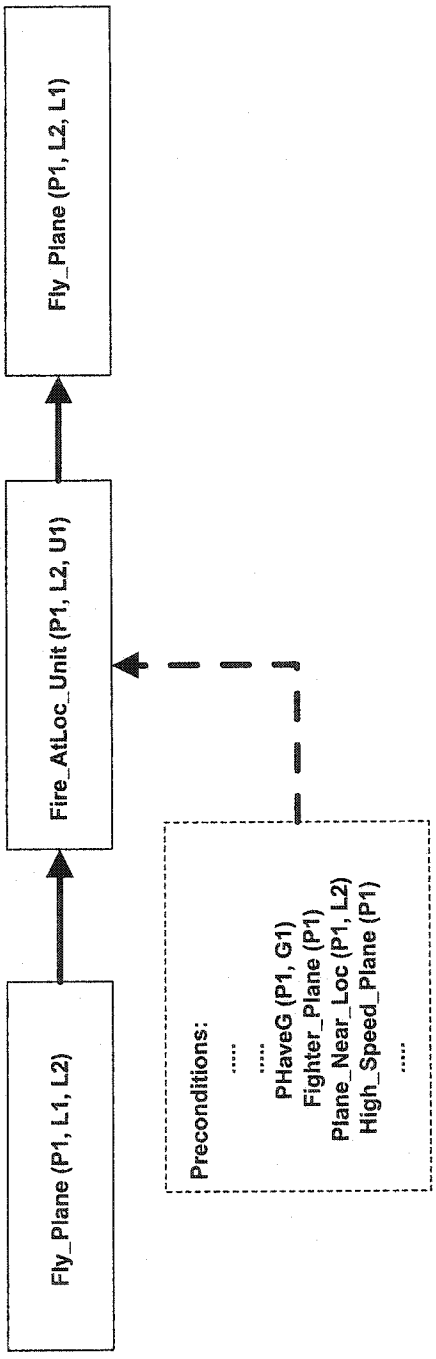


Figure 4.1: A solution in battle-field domain

and *Fighter_Plane*($P1$) might be more important than the others. Because it is possible that for a new problem, there is a plane with high speed and that plane can fly from one location to other, but it does not have a gun and it is not a fighter plane. In that case, *Fire_AtLoc_Unit* action will not be successful, although some of the preconditions are met. So $PHaveG(P1, G1)$ and *Fighter_Plane*($P1$) should be given more weight during the retrieval process of this plan. This is because they might occur as a precondition of other action and it might be difficult to produce these predicates by some other means (repairing).

Weights are also assigned to the goal predicates of the new problem. These weights represent how much "important" or how much "difficult" each goal is. This is valuable in 'any-time' planning. These weights are provided by the user when he gives the new problem specifications for matching and are in an interval of $[0, 1]$.

4.2.3 Weight factor for goal and initial state predicates

Weight factors W_g and W_p represent how much important the goal and initial state similarity measures are, for overall calculation of similarity metric. Goal similarity measure PSM_g and initial state similarity measure PSM_p can be given equal weights i.e. $W_g = W_p = 0.5$ or they can be calculated as follows.

$$W_p = \frac{n_{p_b}}{n_{p_b} + n_{g_a}} \tag{4.1}$$

$$W_g = \frac{n_{g_a}}{n_{p_b} + n_{g_a}} \tag{4.2}$$

4.3 Enhanced CBP Architecture

When a problem-solution pair is retrieved from the case-base, its solution is adapted to solve the new problem. In the CBP architecture of [29], the retrieved solution is modified by instantiating the solution parameters with the new problem parameters. Then the solution is passed to the repairer for fixing any error using any other planning technique. We have modified this architecture by introducing a similarity assessment phase. This phase assesses the similarity value, computed by similarity metric, by taking a feedback from repairer. For simulation purpose, we have implemented fuzzy plan replayer that replays the solution of the retrieved case for computing the repair effort. Then the output of replayer is evaluated by similarity assessment phase. In actual this effort will be given as a feedback from repairer. The modified CBP architecture is shown in figure 4.2.

4.3.1 Plan replaying using fuzzy inference system

In planning, each action has a set of pre-conditions and post-conditions (see Fig. 4.3). In crisp logic, replaying a plan step means checking whether all pre-conditions of an action in a plan are satisfied or not. In case of satisfaction of all pre-conditions, that action is applicable in current situation and the post-conditions of that action become true. But in fuzzy planning system, the pre-conditions and post-conditions

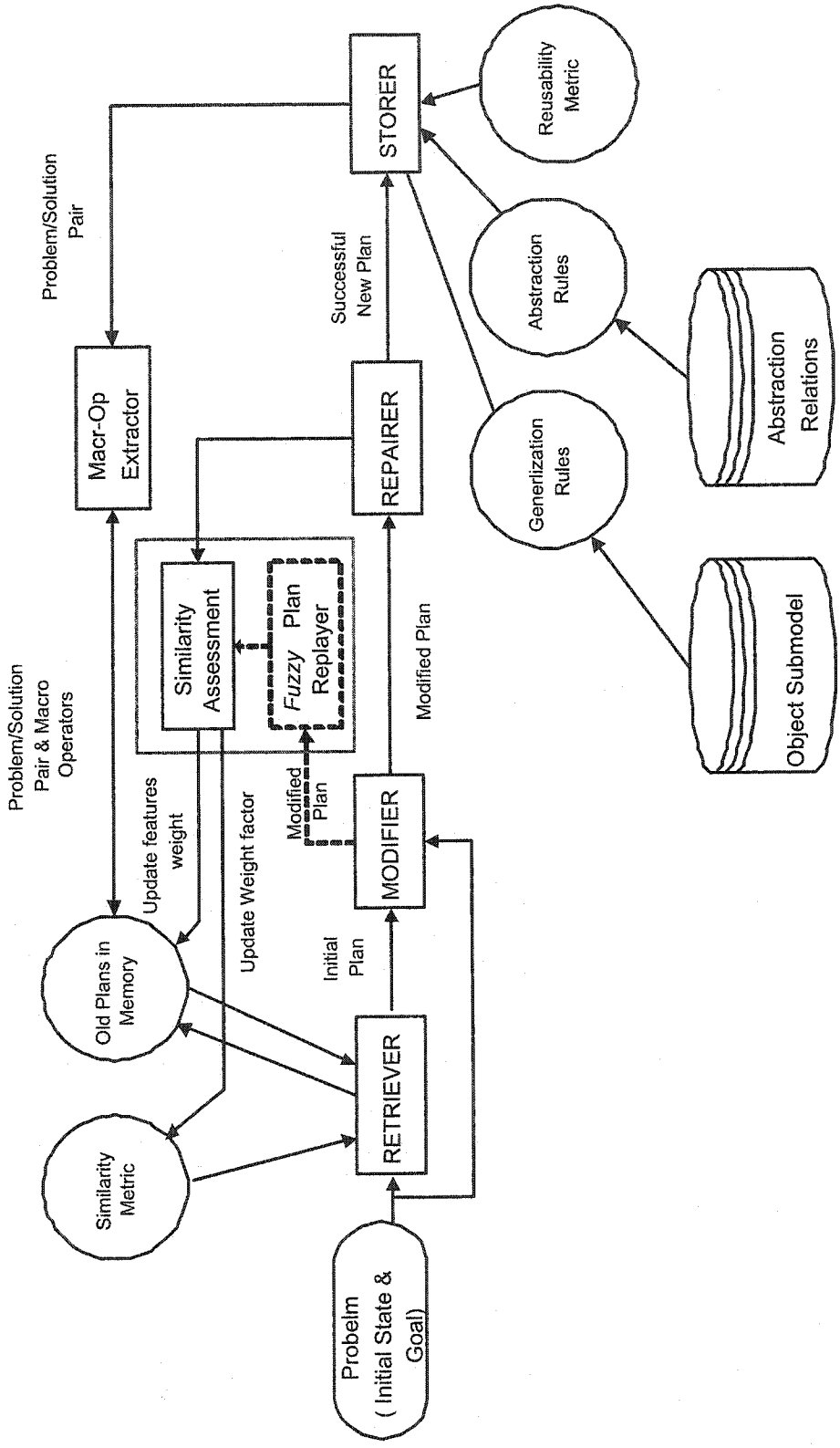


Figure 4.2: Enhanced CBP Architecture

are in the form of fuzzy predicates. These fuzzy predicates are represented by membership functions (we are representing fuzzy predicates using triangular membership function). And these fuzzy predicates have a degree of truth as compared to completely true or false value of crisp predicates. So for fuzzy plan replaying a fuzzy inference system is required in contrast with two-valued logic inference system. In [18], a complete plan replay system is implemented for crisp predicates based planning. We are using the same idea for fuzzy plans by using fuzzy inference system.

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. We have implemented a complete plan replaying system using fuzzy inference engine. To this system, input is given in form of fuzzy predicates with associated triangular membership functions. For applying an action, a fuzzy intersection is taken for all the pre-condition fuzzy predicates of that action with their corresponding fuzzy predicates in the new problem. By this way we find the degree to which each new problem fuzzy predicate has been satisfied for this action. Next, we apply fuzzy AND operation with *min* method on all of the preconditions of that action. The output of this operation is a single truth value. This truth value is used as input for implication process whose output is a fuzzy set. For an action to be applicable, this truth value should be greater than a certain threshold value required for that action. That is, as each action has an associated threshold value A_{thr} which must be satisfied for applying that action. It is different from crisp logic, where an action is applicable when all of its preconditions exist. Here, the existence of all pre-conditions is required with a minimum truth value

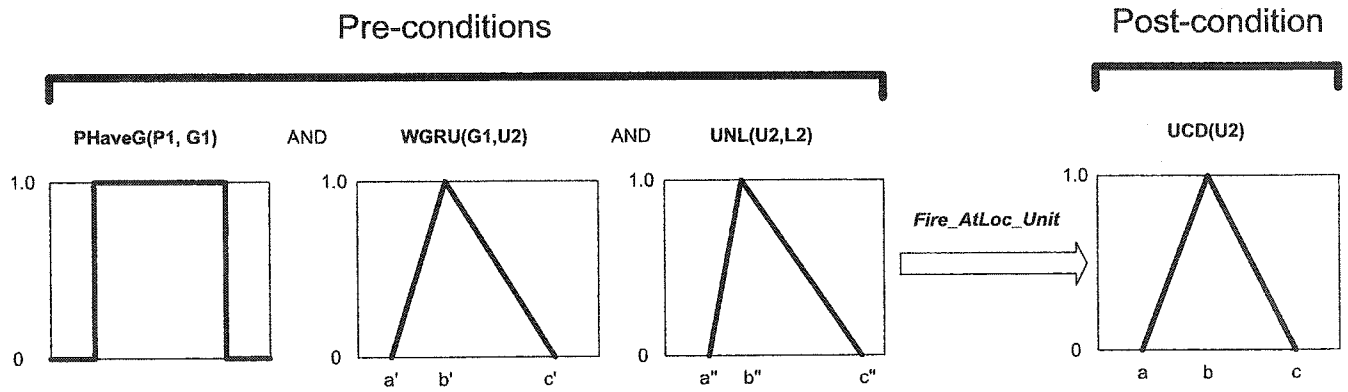


Figure 4.3: An example action *Fire_AtLoc_Unit with pre-conditions and post-conditions*

greater than A_{thr}. We are using *min* implication method in implication process. The result of this implication are truncated postconditions fuzzy sets, which show how much true the postconditions are after applying an action. For plan replay simulation, these postconditions are added in the current state of the problem with their corresponding membership values. The implication process is illustrated for *Fire_AtLoc_Unit action* in figure 4.4.

In figure 4.4, after applying 'AND' operator on pre-conditions, we get 0.8 as the truth value of the whole operation. This value is checked against the threshold value of action *Fire_AtLoc_Unit* action. In this case truth value is assumed to be greater than the action threshold. Hence the action is applicable, so implication operator is used for truncating the post-condition *UCD(U2)*, that gives a membership value of 0.8. For plan replay simulation, this post-condition will be assumed true with a membership value of 0.8.

4.3.2 Plan Replay Evaluation and Weight Adjusting

The replay of a plan will be either successful or unsuccessful. The idea for evaluating replay is similar to [18] in its simplified form with some modification. Instead of using derivational plan replay [27], we have implemented standard replay. In this kind of replay, parameters (objects) of actions and predicates of a case are substituted with appropriate problem parameters (objects). The criterion for updating the feature weights is as follows; if in the retrieved case (problem-solution pair) the percentage of successfully replayed actions is less than a certain threshold then the replay, and

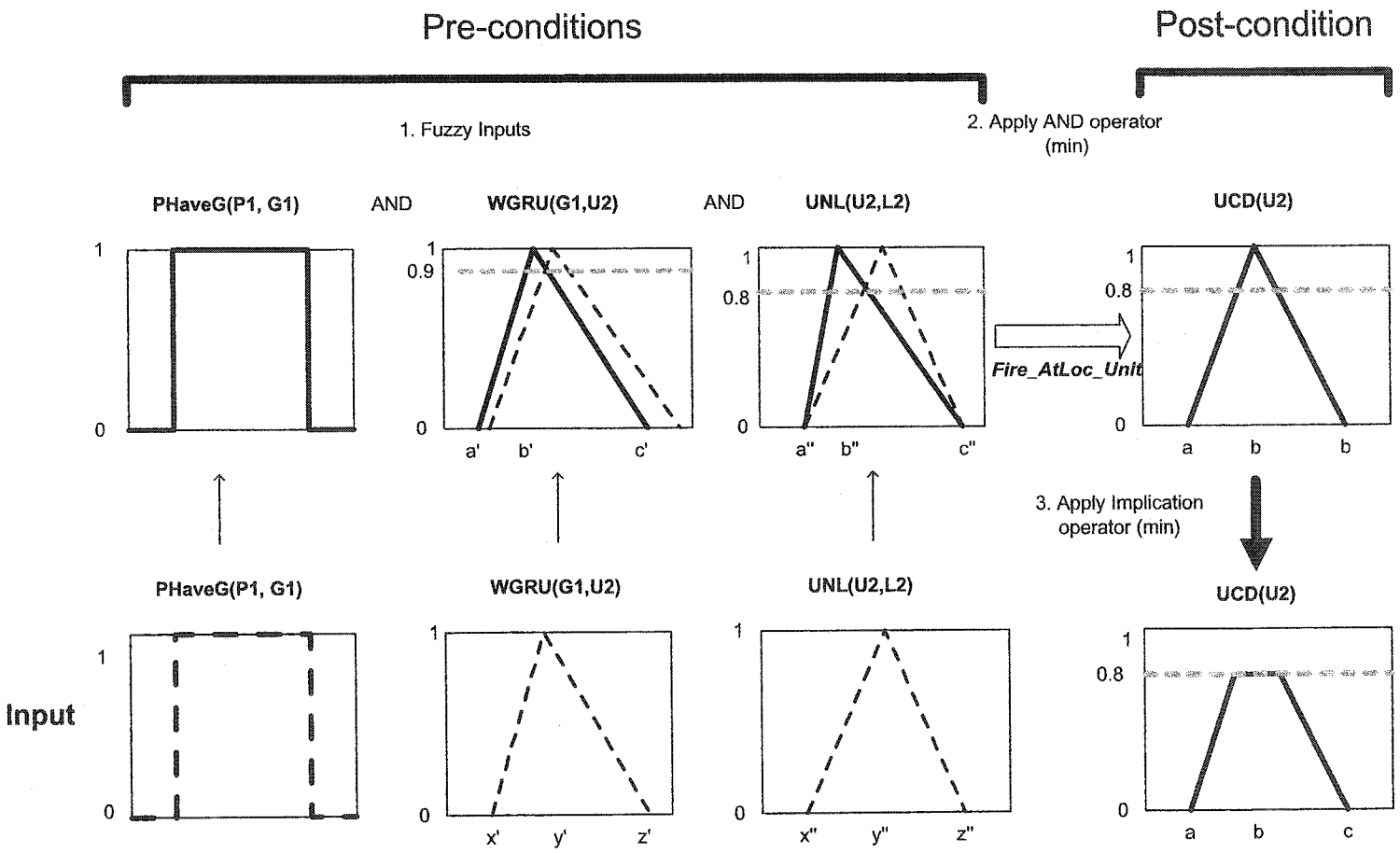


Figure 4.4: An example implication process for *Fire_AtLoc_Unit action*

hence the retrieval, will be considered as failure. In case of missing any desired goal, repair effort is also estimated. This effort shows how difficult a goal is to achieve or in other words how many actions are required to achieve a missing goal. Although the actual effort will be computed by repairer phase yet for simulation purpose, we assume that every missing goal costs one action. Depending upon the output of this plan replay simulation, similarity assessment phase updates the weights of the initial state predicates of a case.

If the retrieval of the case is failed, the weights of those predicates which were missing in the new problem and were present in the case are increased. This is because their absence was important that is why all plan steps were not replayed successfully. On the other hand, if the retrieval of a case is successful, the weights of those predicates which were missing in the new problem and were present in the case are decreased. The reason for decreasing the weights of these predicates is that their absence was not important that is why most of the plan steps were replayed successfully.

It is worth mentioning that the same similarity assessment phase can be used for updating the weight factors of goals W_g and initial state W_p. However, this needs more investigation with some kind of heuristic.

4.4 Weighting Model

It has been shown in [82],[83] and [84], that weighting predicates (features) can improve accuracy in case-base reasoning(CBR), case-based planning(CBP) and machine learning. For updating the weights of initial state predicates, we have used the idea of introspective learning. In this technique, the predicates´ weights are increased or decreased on the basis of problem solving performance.

For updating the weights of predicates a constant value can be used or it can decay as the learning proceeds. We are using decay policy mentioned in [84]. The formulas for increasing and decreasing predicates´ weights are as follows:

$$w_i(t+1) = w_i(t) + \triangle_i \frac{F_c}{K_c} \qquad for\ Failure,$$

$$w_i(t+1) = w_i(t) - \triangle_i \frac{F_c}{K_c} \qquad for\ Success,$$

where K_c indicates the number of times a case has been successfully retrieved and F_c reports the number of times a case has been incorrectly retrieved. This means that when a case is successfully retrieved K_c is incremented by 1 and if the retrieval of a case is unsuccessful F_c is incremented by 1. We have modified this update in K_c and F_c as follows; when a case is replayed, the increment in K_c is the % of plan successfully replayed R_s and increment in F_c is $1 - R_s$. This helps in differentiating a case with 75% successful plan replay and a case with 90% successful plan replay, when both are considered successfully replayed plans. The ratio $\frac{F_c}{K_c}$ reduces the

influence of the weight update as the number of successful retrievals increases. The value $\triangle_i$ determines the initial weight change. We tested different values of $\triangle_i$ for updating the weights. Note that F_c and K_c belong to the whole case and not to the predicate itself.

4.5 The Modifier: Instantiating Solution parameters

In CBP architecture of [29] the modifier is responsible for modifying the retrieved solution by instantiating the solution parameters with the new problem parameters. Instead of finding similarity value for each substitution, as done in [25], we have used Tabu Search (TS) and Simulated Annealing (SA) algorithms for instantiating the solution parameters. An illustrative example for instantiating the solution parameters with the new problem parameters is given in section 4.6.

4.5.1 Tabu Search (TS) for maximizing Similarity Value

Tabu search is an iterative heuristic that has been applied for solving a range of combinatorial optimization problems in different fields [85]. Tabu search starts from an initial feasible solution and carries out its search by making a sequence of random moves or perturbations. A *tabu list* is maintained to store the attributes of a number of previous moves. This list prevents bringing the search process back to already visited states. In each iteration, a subset of *neighbor* solutions is generated by making

a certain number of moves and the best move (the move that resulted in the best solution) is accepted, provided it is not in the tabu list. Otherwise, if the said move is in the tabu list, the best solution is checked against an *aspiration criterion* and if satisfied, the move is accepted. Thus, the aspiration criterion can override the tabu list restrictions. It is desirable in certain conditions to accept a move even it is in the tabu list, because it may take the search into a new region due to the effect of intermediate moves.

The behavior of tabu search heavily depends on the size of tabu list as well as on the chosen aspiration criterion. Different sizes of tabu list result in short-term, intermediate term, and long-term memory components that can be used for intensifying or diversifying the search. The aspiration criterion determines the extent to which the tabu list can restrict the possible moves. The structure of TS is given in Figure 4.5. The detailed description of tabu search can be found in [85].

TS uses single solution and tries to optimize it with iterations. The unique feature of TS is its *memory element*, that is used to record some characteristics of a certain number of previous moves. This feature is realized by the use of *Tabu list*. The number of moves whose characteristics can be recorded depends on the size of this list. This feature prevents the search process from cycling (i.e. revisiting a point) in the search space. The details of various steps are presented in following sub-sections.

Initialization and Similarity Evaluation

The initialization step depends on the number of different objects present in a case and the new problem. Let P_{obj} be the a set of objects in the new problem. Let C_{obj} be a set of objects in a case. Now if the $|C_{obj}| \geq |P_{obj}|$ then initialize all P_{obj} with first C_{obj}, leaving the remaining C_{obj} in case of $C_{obj} > P_{obj}$. On the other hand, if the $|P_{obj}| > |C_{obj}|$ then initialize all C_{obj} with first P_{obj}, leaving the remaining P_{obj}. Next the similarity is evaluated using the similarity metric and is stored as the best solution cost.

Neighborhood Generation

In each iteration, a number of neighbors of the current solution are generated by making perturbations, such that two random numbers are generated between 1 and

Algorithm *Tabu Search*;
(*Ω is the set of feasible solutions*)
(*S is the Current solution*)
(*S^* is the Best solution*)
(*$Sim_{Overall}$ is the Similarity Value*)
(*$N(S)$ is the neighborhood of S*)
(*V^* is the sample of N(S)*)
(*TL is the Tabu list*)
(*AC is the Aspiration Criteria*)
Begin
For all cases in the case base **Do**
Start with an initial feasible solution;
Initialize TL and AC
For Fixed number of Iterations **Do**
Generate Neighbor solutions V^* by randomly swapping two objects.
Find best $S^* \epsilon N(S)$;
If move S to S^* is not in TL **Then**
Accept move and update S^*;
Update TL and AC;
Else
If $Sim_{Overall}$ > AC **Then**
Update TL and AC;
End If
End If
End For
End For
End(*of *Tabu Search*)

Figure 4.5: Tabu Search for maximizing Similarity Value by instantiating fuzzy predicates

N, where N is the maximum number of objects present either in the case or in the new problem.

The quality of the final solution generated by TS depends on the number of neighbor solutions generated. Too small number of neighbor solutions prevents TS from searching the solution space effectively. On the other hand, too large number of neighbor solutions costs excessive run time without providing a significant improvement in the quality of solution. Therefore, the number of neighbor solutions to be generated in each iteration, needs to be chosen after a careful observation of the underlying problem's nature and size.

Tabu List and Aspiration Criterion

Tabu list introduces memory element in search process. Its purpose is to avoid revisiting a point (solution) in the search space. This is implemented by storing some characteristic of a certain number of previously accepted moves.

Implementation of tabu list requires two decisions to be made. First, what characteristic of the move should be stored in tabu list. This decision has a significant effect on search quality as well as memory requirements of TS. The chosen characteristic should identify the move, so that it can accurately be used to restrict the corresponding move and to consequently fulfill the purpose of tabu list. In the present implementation, the characteristic of the move stored in tabu list is the index of object.

Second decision that needs to be made is regarding the size of tabu list. This

decision affects time and space requirements of TS.

The aspiration criterion used is following: if the best neighbor solution of the current iteration is better than the global best solution, then tabu list restrictions are overridden and the solution is accepted. Also the global best solution and the aspiration criterion is updated.

Stopping Criterion

TS was run for different number of total iterations, and depending on experimental observations, it was found that number of iterations are dependent on the number of objects present in the problem/case.

4.5.2 Simulated Annealing (SA) for maximizing Similarity Value

Simulated annealing algorithm is based on the phenomenon of the cooling of a metal. A metal is heated at very high temperature, then cooled down in a controlled manner so as to achieve a minimal energy state. The similar concept is applied to determine the optimal solution to NP- hard problems.

SA starts with initialization of some parameters with include cooling rate α, and number of iterations the algorithm is supposed to run. The initial temperature is also calculated at the start of Algorithm.

The metropolis function is important issue in this algorithm. At a particular temperature the predefined number of moves are made. The function accepts the

Algorithm *Simulated Annealing*$(S_0, T_0, \alpha, \beta, M, Maxtime)$;
(*S_0 is the initial solution*)
(*BestS is the best solution*)
(*T_0 is the initial temperature*)
(*α is the cooling rate*)
(*β a constant*)
(*$Maxtime$ is the total allowed time *)
(*M is the time until the next parameter update*)
Begin
$T = T_0$;
$CurS = S_0$;
$BestS = CurS$; /*$BestS$ is the best solution seen so far*/
$CurCost = Cost(CurS)$;
$BestCost = Cost(BestS)$;
$Time = 0$;
Repeat
Call $Metropolis(CurS, CurCost,$
$BestS, BestCost, T, M)$;
$Time = Time + M$;
$T = \alpha T$;
$M = \beta M$;
Until $(Time \geq Maxtime)$;
Return$(BestS)$
End.(*of *Simulated Annealing*)

Figure 4.6: Simulated Annealing for maximizing Similarity Value by instantiating fuzzy predicates.

```
Algorithm Metropolis(CurS, CurCost,
                     BestS, BestCost, T, M);
Begin
    Repeat
        NewS = Perturb(CurS);
        /*by adding or removing a link*/
        NewCost = Cost(NewS);
        Gain = (NewCost − CurCost);
        If Gain > 0 Then
            CurS = NewS;
                If NewCost > BestCost Then
                    BestS = NewS;
                    BestCost = NewCost;
                EndIf
        Else
                If (RANDOM < e^(−Gain/T)) Then
                    CurS = NewS;
                    CurCost = NewCost;
                EndIf
        EndIf
        M = M − 1;
    Until M = 0
End (*of Metropolis*)
```

Figure 4.7: The Metropolis procedure.

good moves readily, and bad moves are accepted with some probability.

Initial Solution

Initial solution is required as a starting point in simulated annealing algorithm the method of generation of initial solution is the same as discussed in Section 4.5.1 for TS.

Neighborhood Generation

A transition mechanism generates a new solution (Schedule) $S$from a current solution (Schedule) S. The neighbor solution is generated with the method as discussed in Section 4.5.1 for TS .

Cooling Schedule

Simulated annealing (SA) converges to a global optimum if an infinite number of transitions are allowed at each temperature T_i and the temperature approaches to zero as the time approaches to infinity. However, the number of each iterations at each temperature T_i and the number of different temperature steps are usually finite in real implementation. Therefore SA can not find the optimal solution all the time. The three important issues which need to be considered for cooling policy are:

1. the initial temperature value.

2. the final temperature value or the stopping criteria (condition).

3. the decrement rule of the temperature of the number of temperature steps.

Initial Temperature

Initial temperature should be set that almost all possible transitions at the initial temperature T_o are accepted, i.e. $e^{-\frac{\delta C}{T_o}} \approx 1$. This is approximated by defining the *acceptance ratio* χ, which is obtained by dividing the number of accepted trial solutions by total number of trial solution. N neighbor solution are generated on the trial solution (sequence).

Final Temperature

If the temperature in annealing process drops below 1×10^{-6} then it is kept constant at this level for the remaining iterations.

Decrement rule of temperature

The decrement rule used in our experimentation was as follows

$$T_{i+1} = \alpha T_i$$

Where $i \geq 0$, $0 < \alpha < 1$, T_i is the current temperature and T_{i+1} is the new temperature to be set from the current temperature by decrement rule.

4.6 Illustrative Example

Following is an illustrated example for instantiating/mapping between a case and a new problem for logistic transportation domain. The objects, footprint/initial state predicates and goal predicates are shown below for both case and a new problem.

Old problem-solution pair (case) in memory:

- *Footprint Predicates:* $LOCAT(F1, C1)$, $LOCAT(F2, C1)$, $ATOBJ(O1, F1)$, $ATTRU(T1, F1)$, $SAMECITY(F1, F2)$, $TINCITY(T1, C1)$
- *Goal Predicates:* $ATOBJ(O1, F2)$

New Problem:

- *Initial State Predicates:* $ATOBJ(O10, F30)$, $ATOBJ(O20, F30)$, $ATTRU(T10, F30)$, $SAMECITY(F10, F20)$, $SAMECITY(F30, F10)$, $SAMECITY(F20, F30)$, $TINCITY(T10, C10)$, $LOCAT(F10, C10)$, $LOCAT(F20, C10)$, $LOCAT(F30, C10)$
- *Goal Predicates:* $ATOBJ(O10, F10)$, $ATOBJ(O20, F20)$

After running TS/SA, the maximum similarity value (0.7045) is found with the following instantiation:

$O10/O1$, $T10/T1$, $F30/F1$, $F10/F2$, $C10/C1$

Note that this was tested with different number of iterations for TS and SA. The same result was achieved by exhaustive search also.

Chapter 5

Experiments and Results

In this chapter, we will discuss the experiments performed using the Fuzzy Predicate-based Dynamic Similarity Metric (FDSM). We will compare the results with some other similarity metrics developed for case-based planning and reasoning systems. Moreover, we will provide a comparison between Tabu Search (TS) and exhaustive search (ES) for instantiating the problem/case fuzzy predicates by modifier. This chapter is organized as follows. Section 5.1 provides some details about the domains used in our experiments. Section 5.2 discusses some validation experiments for FDSM. Section 5.3 shows the results of our experiments for logistic and battle field domain. The effectiveness of FDSM is shown in section 5.4. Section 5.5 gives comparison results of TS and ES for instantiating predicates in modifier step of CBP, which is followed by the conclusion in section 5.6.

5.1 Problem Domain

We performed experiments with the domain of logistic transportation (see Appendix A) and battle field (see Appendix B). The logistic transportation domain is originally specified in [19]. In this domain there are different resources e.g. trucks and airplanes are used for moving objects from one place to another. This domain is specified using predicates which are crisp in nature, i.e. these predicates have values either true or false. This domain is used for providing the comparison between our developed similarity metric and similarity metrics presented in [25] and [18].

The second domain used in the experiments is the battle field domain. In this domain it is rare that all relevant information and outcomes are known with certainty and precision. So, for our experimental purpose, we have developed the specification of this domain using fuzzy predicates. The specifications of this domain are huge in contrast with logistic transportation domain. We have done experiments with around 100 fuzzy/nonfuzzy predicates and 40 actions.

5.1.1 The Symbolic Specification of Domains

In AI planning, a symbolic specification is used to represent the problem domains. A domain can be specified by the of the following components:

- **Objects:** Objects are the basic units for representation. They describe the entities in the world. Example of objects in the domain of battle field are the *Unit, Location, Plane* etc. Symbolically, an object is represented as a string

of characters. For example, U may indicate a *Unit* or P my indicate a *Plane*.

- **Predicates:** Predicates indicate a relations between objects. Example of a predicate in the domain of battle field are $Unit\_FireAt\_Plane(U_1, P_2)$ or $Unit\_Near\_Loc(U_1, L_1)$. In crisp logic a predicate can have value either *True, or False* while fuzzy predicates have a degree of truth in form of membership value. We are using fuzzy predicates in our domain specification instead of simple crisp predicates.

- **Operators:** Operators describe actions, which are the basic units from which plans are made. A widely used representation for operators is the STRIP-operators [1]. These operators are constituted of arguments, constraints, pre-conditions and effects.

5.1.2 Case Representation

For retrieval of similar cases, a case is represented by

- Case identification (number)
- Footprint Predicates
- Post-condition or Goal situations
- Weights for foot-printed predicates
- Failure and success statistics

5.1.3 Problem Representation

A new problem is represented by

- Initial State Predicates
- Post-condition or Goal situations
- Weights for the goals
- Weighting factor

For new problems and cases the pre-conditions and post-conditions are presented by fuzzy sets. We have used triangular membership function to represent the fuzzy predicates in our experiments.

5.2 Validation of FDSM as a Similarity Metric

For a similarity measure to be a metric it must satisfy the following four criteria presented in [86] and [87]:

1. Given two cases j and k, $d_{jk} = d_{kj} \geq 0$, where d_{jk} is the distance between case j and case k.

2. Given three cases j , k and l, $d_{jk} \leq d_{jl}+d_{kl}$ (this is called the triangle inequality or the metric inequality).

3. Given two cases j and k, if $d_{jk} \neq 0$, then j is not identical to k.

4. Given two identical cases j and j', $d_{jj'} = 0$.

	objects	pre-conditions	post-conditions	# of actions
case 1	7	26	3	7
case 2	7	25	2	7
case 3	9	32	3	9
case 4	8	27	2	4

Table 5.1: Example Cases from Battle `Field Domain`

Symmetry	Minimality	Self Similarity
$d_{12} = d_{21}$	$d_{11} \leq d_{34}$	d_{11}
0.326291	$0 \leq 0.268414$	0

(a)

Symmetry	Minimality	Self Similarity
$d_{23} = d_{32}$	$d_{22} \leq d_{13}$	d_{22}
0.400094	$0 \leq 0.152267$	0

(b)

Symmetry	Minimality	Self Similarity
$d_{34} = d_{43}$	$d_{33} \leq d_{12}$	d_{33}
0.248146	$0 \leq 0.268414$	0

(c)

Table 5.2: Experimental Validation

Note that in above criteria distance d between two cases j and k is $d_{jk} = 1 - Similarity_{jk}$. It has been stated in [87] that 1, 3 and 4 can be proved by experiments but triangular inequality is not easy to prove experimentally. For validating our similarity measures as a metric, some experiments were performed in both logistic transportation and battle field domains. Table 5.1 shows some example cases selected for battle field domain experiments and table 5.2 shows experimental validations of 1, 3 and 4.

5.3 Results and discussion

In this section we'll discuss the experiments performed in logistic and battle field domains for comparing FDSM with other similarity metrics presented in [18], [25], [78] and [75].

5.3.1 Experiment Setup

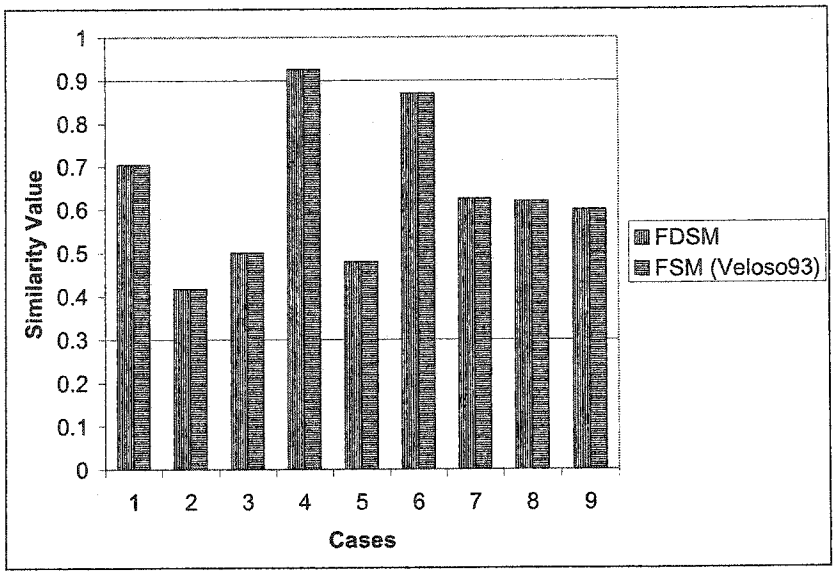
For each domain we developed a case base for storing the problem-solution pairs. Logistic Transportation domain was constructed with crisp predicates and actions specified in [19]. While for battle field domain we used the fuzzy predicates and actions specified in Appendix B. We used MS. Access Database for storing these predicates, actions and problem-solution pairs.

5.3.2 Performance of FDSM in Logistic Transportation Domain

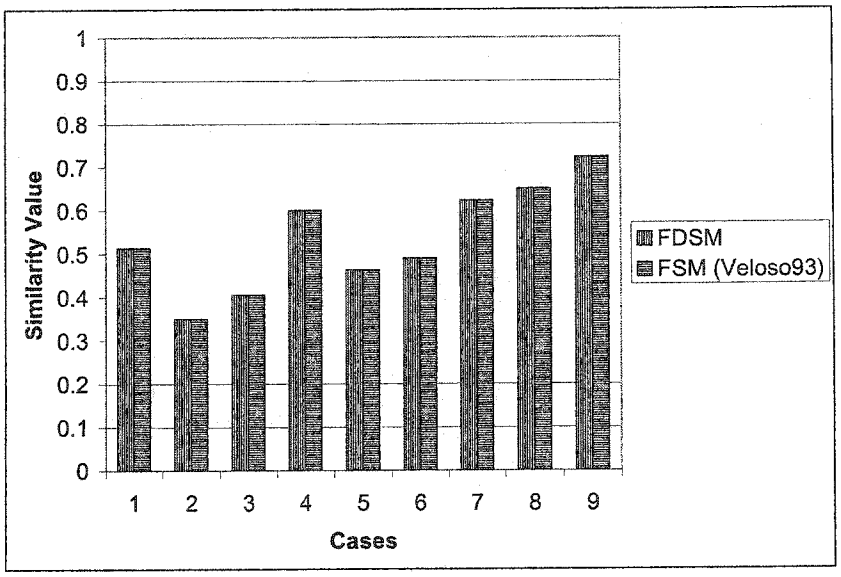
We performed experiments on different problems with different number of goals from logistic transportation domain and found consistency between our similarity metric and the similarity metrics presented in [18] and [25]. The similarity metrics presented in [18] and [25] use crisp predicates for representing cases. The experiments show that our similarity metric is also applicable for crisp predicate based domains.

Comparison between FDSM and Other footprint-based similarity metrics

Different experiments were performed for comparing FDSM with metrics of [18] and [25]. The similarity metric presented in [25] is a static footprint-based metric (FSM) in which all features (predicates) have same importance (weight). The similarity metric presented in [18] uses feature weighting and is an extension of [25] as it also uses footprint predicates for matching. Our similarity metric is consistent with both of these metrics and in addition it performs better in situations where user prefers some goals over others (see Table 5.3). This is because goal importance value is also taken into account and incorporated in FDSM. Figure 5.1 shows a comparison of footprint-based similarity metric and FDSM for two problems from logistic domain. Figures 5.2 shows a comparison of weighted footprint-based similarity metric (WFSM) and FDSM.

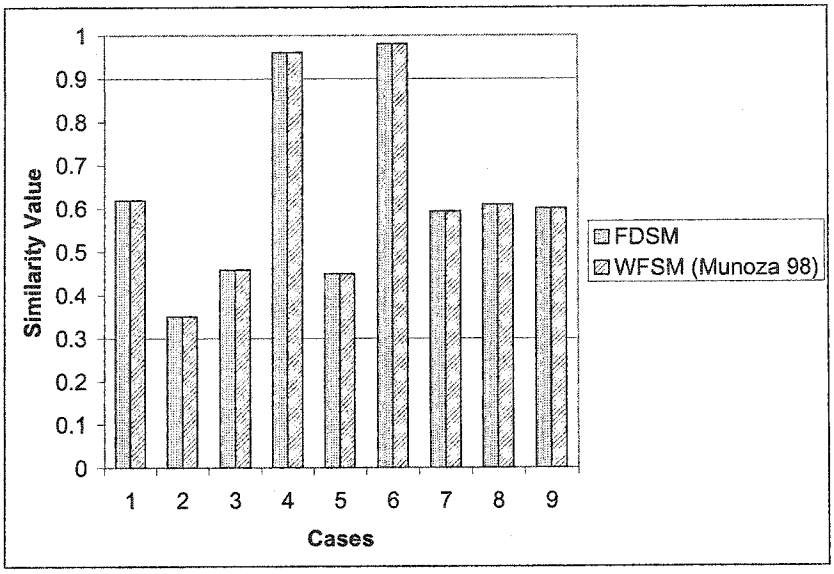


(a)

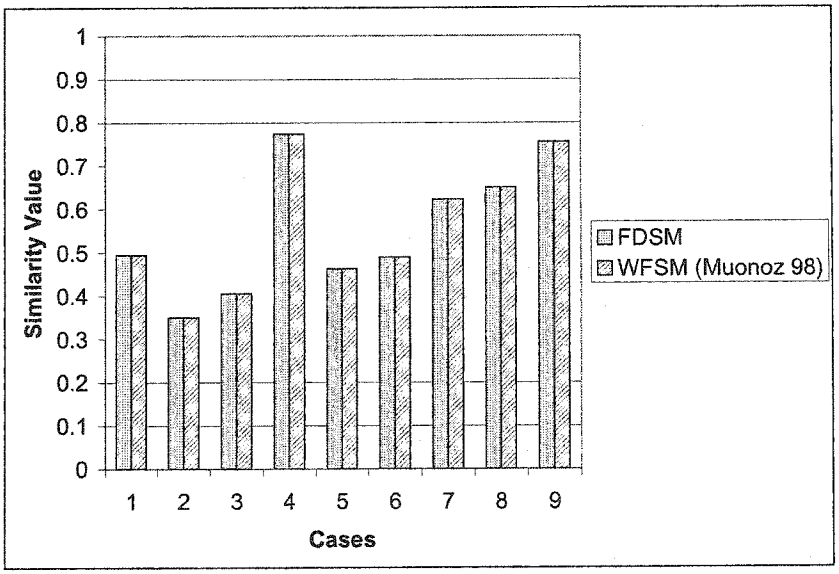


(b)

Figure 5.1: Similarity Value Comparison between FDSM and FSM



(a)



(b)

Figure 5.2: Similarity Value Comparison between FDSM and WFSM

Goal	wt. of Goal
G1	0.25
G2	0.25
G3	0.25
G4	0.25

(a) Problem 1: equal goal wts.

Case #	Sim. Value	Goals Matched
7	0.665323	G_1, G_2
8	0.675	G_3, G_4
9	0.697581	G_2, G_4

(b) Retrieved Cases with $Threshold \geq 65\%$

Goal	wt. of Goal
G1	0.10
G2	0.40
G3	0.20
G4	0.30

(a') Problem 1': diff. goal wts.

Case #	Sim. Value	Goals Matched
8	0.65	G_3, G_4
9	0.722581	G_2, G_4

(b') Retrieved Cases with $Threshold \geq 65\%$

Table 5.3: Effect of Goal weights on retrieval

5.3.3 Performance of FDSM in Battle Field Domain

We performed experiments on different problems from battle field domain for checking the performance of FDSM. The results are compared with two other similarity metrics presented in [78] and [75].

Experiments without learning features´ weights

Initially we performed experiments by statically computing the weights of initial state predicates of all cases in the case-base. These weights just show the contribution of each predicate to the whole plan of a case. Table 5.4 shows the number of predicates and objects in these problems. Table 5.5 shows the number of cases retrieved by each of the similarity metrics and the plan replay result of these cases for 4 problems. Table 5.6 through 5.9 show a comparison between FDSM with

Prob. #	Objects	Pre-conditions	Goals
1	7	26	2
2	13	42	4
3	7	26	3
4	14	53	5

Table 5.4: Example Problems from Battle Field Domain

$W_p = W_g = 0.5$, and W_p and W_g calculated by equations 4.1 and 4.2 respectively. For theses problems, we replayed all the case plans so that we could find that percent-age of missing successful case plans by each metric. This percentage is also shown in the tables. The details of these problems are given in appendix C. The comparison between similarity values for these problems is shown in figure 5.3 through 5.6.

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	6	33.33	66.66	0
MinMax[75]	2	50	50	50
ChenFT[78]	8	25	75	0

(a) Problem 1

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	4	50	50	0
MinMax[75]	0	0	0	100
ChenFT[78]	1	100	0	50

(b) Problem 2

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	4	75	25	0
MinMax[75]	1	100	0	66.66
ChenFT[78]	1	100	0	66.66

(c) Problem 3

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	5	20	80	0
MinMax[75]	0	0	0	100
ChenFT[78]	2	50	50	0

(d) Problem 4

Table 5.5: Experiments without learning feature weights, $X = W_p = W_g = 0.5$, $Retr\_Thr = 75\%$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	6	33.33	66.66	0
FDSM $W_p(4\ .1), W_g\ (4.2)$	5	40	60	0

(a) $Ret\_Thr \geq 75\%$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	3	66.66	33.33	0
FDSM $W_p(4\ .1), W_g\ (4.2)$	3	100	0	0

(b) $Ret\_Thr \geq 80\%$

Table 5.6: Problem 1 : Experiments without learning feature weights, $X = 0.5$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	4	50	50	0
FDSM $W_p(4\ .1), W_g\ (4.2)$	5	40	60	0

(a) $Ret\_Thr \geq 75\%$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	0	0	0	100
FDSM $W_p(4\ .1), W_g\ (4.2)$	2	100	0	0

(b) $Ret\_Thr \geq 80\%$

Table 5.7: Problem 2 : Experiments without learning feature weights, $X = 0.5$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	4	75	25	0
FDSM $W_p(4\ .1), W_g\ (4.2)$	3	100	0	0

(a) $Ret\_Thr \geq 75\%$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	0	0	0	100
FDSM $W_p(4\ .1), W_g\ (4.2)$	3	100	0	0

(b) $Ret\_Thr \geq 80\%$

Table 5.8: Problem 3 : Experiments without learning feature weights, $X = 0.5$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	5	20	80	0
FDSM $W_p(4\ .1), W_g\ (4.2)$	5	20	80	0

(a) $Ret\_Thr \geq 75\%$

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM $W_p = W_g = 0.5$	3	33.33	66.66	0
FDSM $W_p(4\ .1), W_g\ (4.2)$	2	50	50	0

(b) $Ret\_Thr \geq 80\%$

Table 5.9: Problem 4 : Experiments without learning feature weights, $X = 0.5$

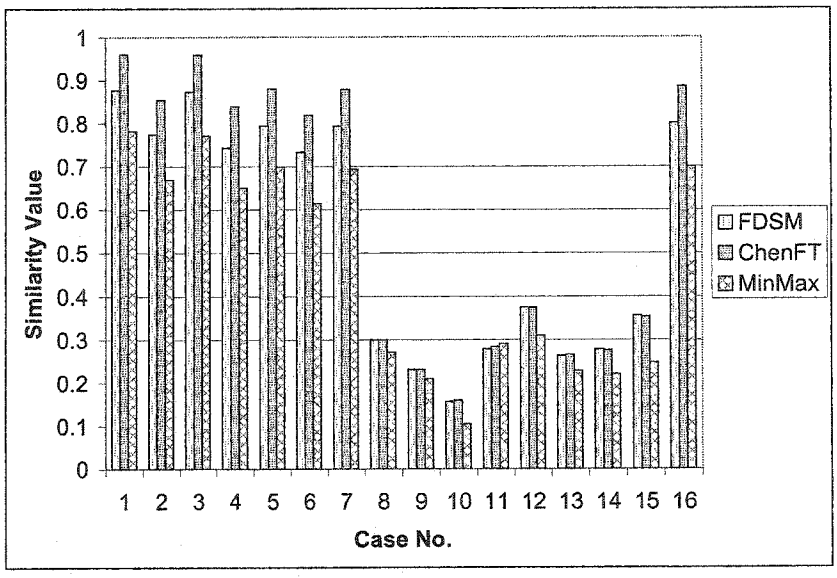


Figure 5.3: Similarity Values for Problem1

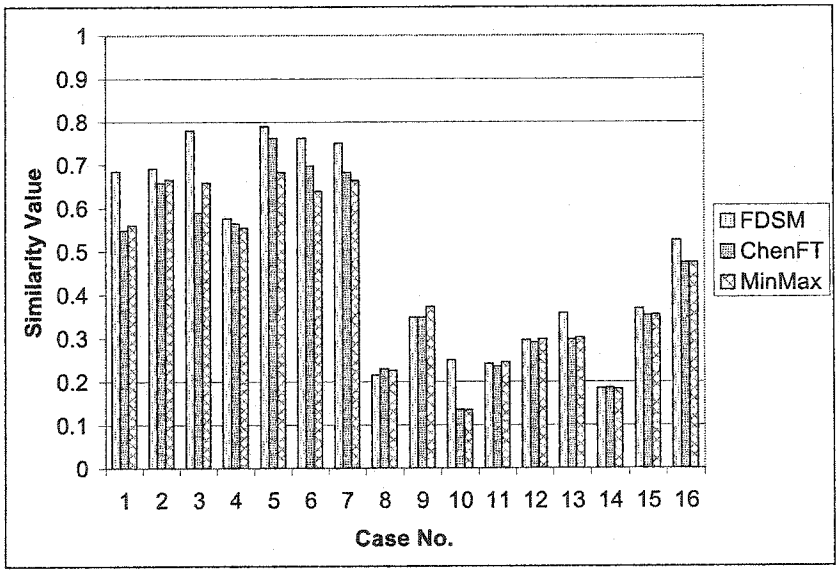


Figure 5.4: Similarity Values for Problem2

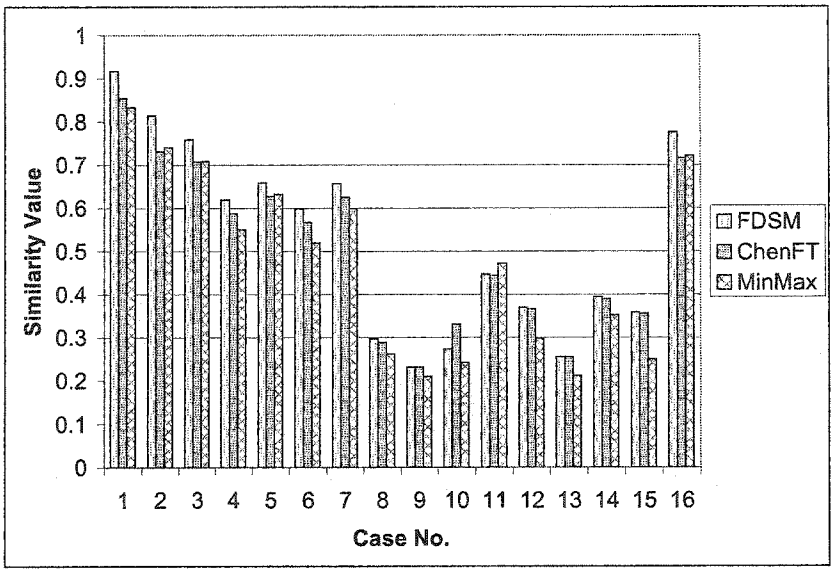


Figure 5.5: Similarity Values for Problem3

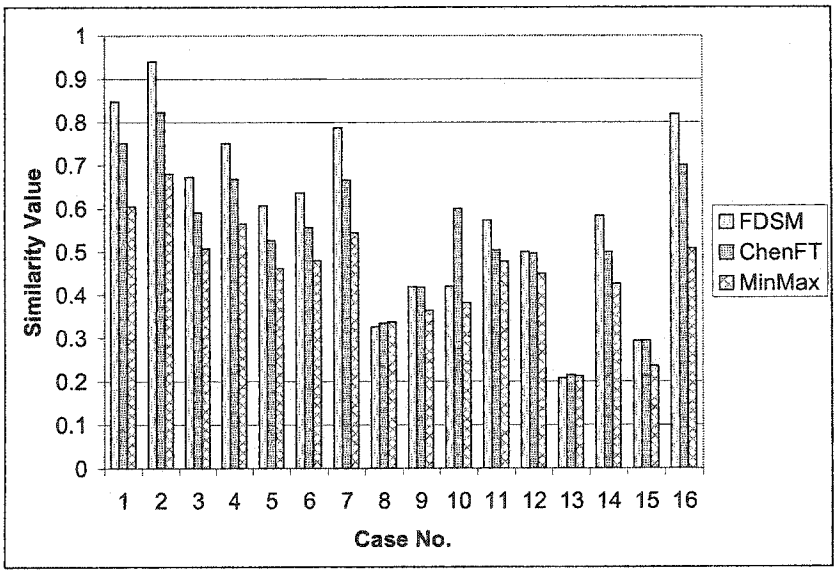


Figure 5.6: Similarity Values for Problem4

Experiments after updating features´ weights

As mentioned in section 4.3.1, the weights of features are updated on the basis of feedback from repairer. In our experiments, after replaying the cases for different training problems, the feature weights were updated appropriately. Note that only those cases were replayed which had similarity value more than the retrieval threshold. These experiments show that the number of retrieved cases are reduced as the features weights are adjusted. There is no effect of updating the features weights on $MinMax$ similarity measure, as it does not take into account the weights in similarity computation. The same problems of previous section were used in these experiments. Table 5.10 shows the number of cases retrieved by each of the similarity metrics and the plan replay result of these cases for these problems after updating the features weights. The comparison between similarity values for these problems after updating feature weights is shown in figure 5.7 through 5.10. The effect of plan replays after executing some training problems is shown in figures 5.19 through 5.20.

It was also found that sometimes no goal is matched between a problem and a case but the plan can be replayed successfully for that case. For taking into account such cases we introduced goal and initial state weighting factors i.e. W_g and W_p in our similarity metric. When the value for these factors are changed, it effects the retrieval of cases. Table 5.11 shows one example problem for which no case was retrieved when $W_g = W_p = 0.5$ and $Thr = 75\%$ for FDSM, as no goal was

matched. But when we changed $W_g = 0.3$ and $W_p = 0.7$ a case was retrieved for same threshold and it was replayed successfully. While other two similarity metrics (MinMax and ChenFT) were failed in retrieving any similar case. The effect of different values of W_g, Wp and X is shown in 5.11 through 5.16 for one problem.

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	2	100	0	0
MinMax[75]	2	50	50	0
ChenFT[78]	7	28.57	71.42	0

(a) Problem 1

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	2	100	0	0
MinMax[75]	0	0	0	100
ChenFT[78]	1	100	0	50

(b) Problem 2

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	3	100	0	0
MinMax[75]	1	100	0	66.66
ChenFT[78]	2	100	0	33.33

(c) Problem 3

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	1	100	0	0
MinMax[75]	0	0	0	100
ChenFT[78]	1	100	0	0

(d) Problem 4

Table 5.10: Experiments after updating feature weights

	Case Retrieved	% of Successful Replay	% of Failed Replay	% of Missing Successful Plans
FDSM	1	100	0	0
MinMax[75]	0	0	0	100
ChenFT[78]	0	0	0	100

Table 5.11: Problem 5 : W_g= $0.3, W_p = 0.7, X = 0.5$

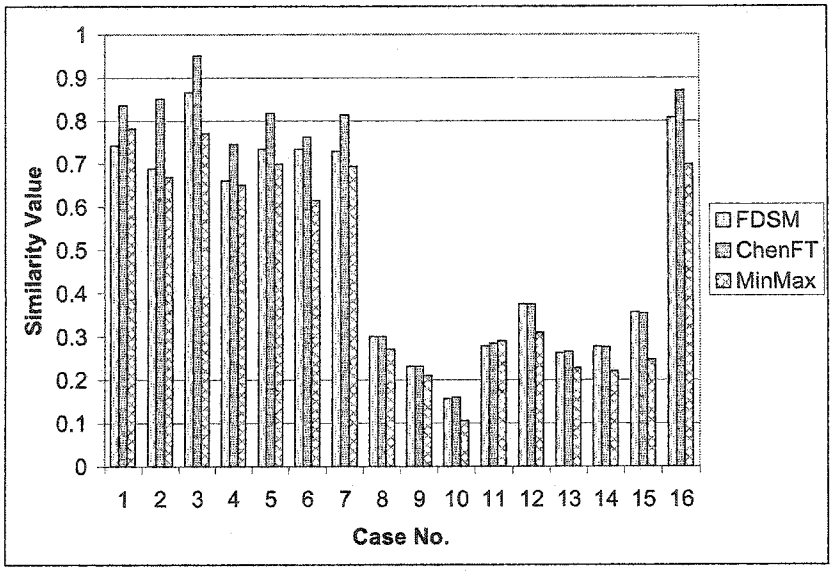


Figure 5.7: Similarity Values for Problem1

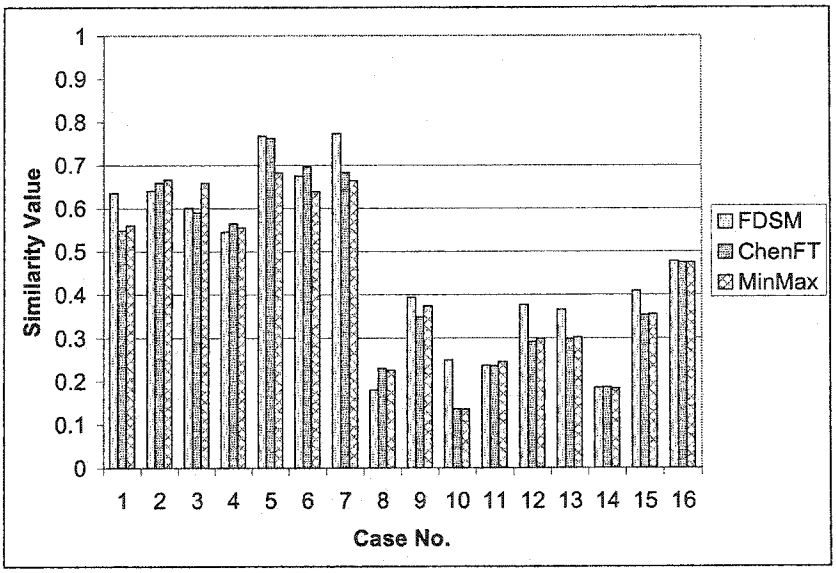


Figure 5.8: Similarity Values for Problem2

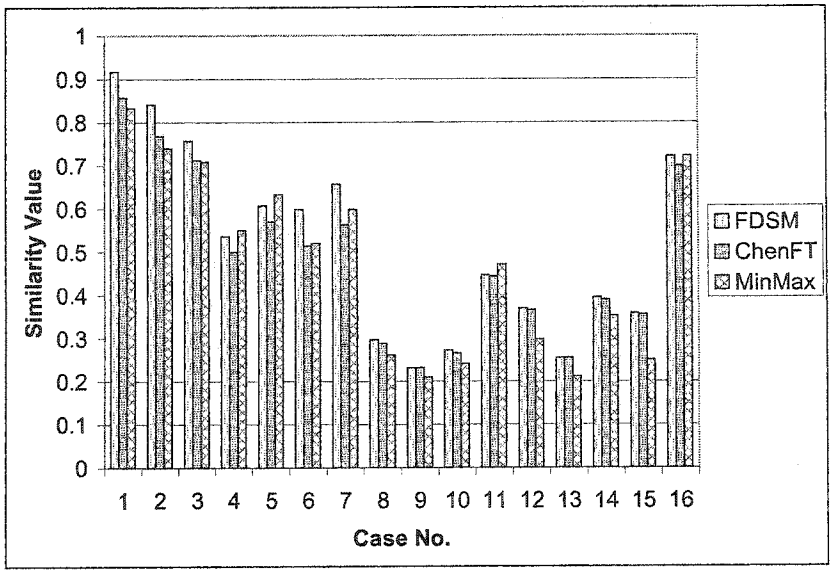


Figure 5.9: Similarity Values for Problem3

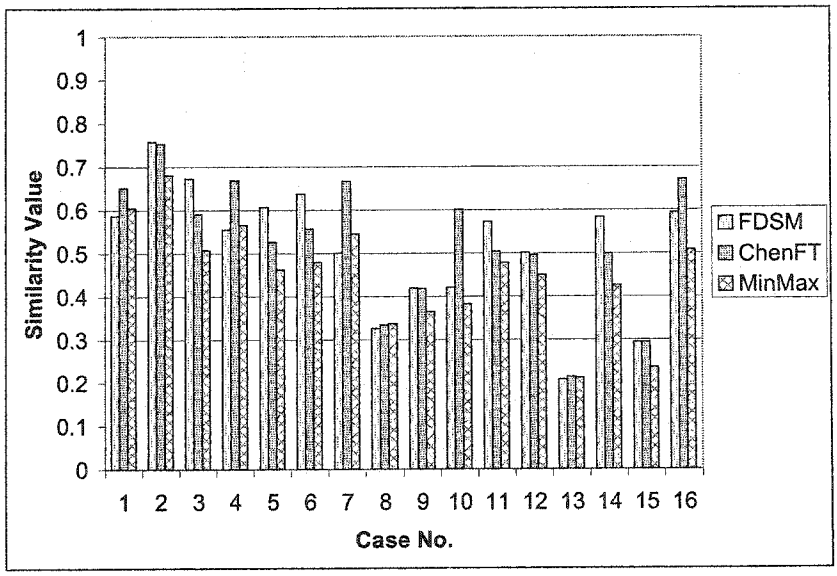


Figure 5.10: Similarity Values for Problem4

5.4 Soundness of FDSM

For confirming that our developed similarity metric is effective and sound, we correlate the computed similarity value between a problem and a case with the percentage of plan replay for that case. It was found that if the similarity value between a problem and a case was high then the percentage of plan replay for that problem/case was also high. This trend was found largely consistent for different problems. We used equations 4.1 and 4.2 for calculating the values of $W_p$and W_g respectively for these experiments. The figures 5.21 through 5.25 show this trend.

5.5 Comparison of Exhaustive Search, TS and SA for Instantiating Fuzzy Predicates

We used Tabu Search (TS) and Simulate Annealing (SA) algorithms for instantiating the fuzzy predicates appropriately, so that we could find the maximum similarity value. In [25] each problem object is replaced with each case object for finding the maximum similarity. This can work only for very small number of objects. Figure 5.26 shows the time for solving problems with different number of objects using exhaustive search and the time for solving same problems by using TS. A comparison between exhaustive search and TS time for solving different problems with their similarity values is shown in Table 5.12. It was also noted that time not only depends on number of objects but also on the number of predicates of a case and

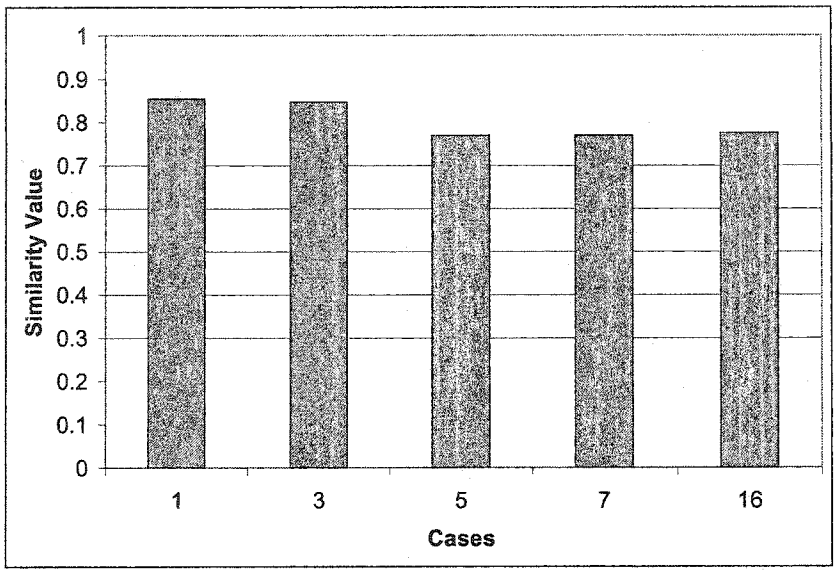


Figure 5.11: Similarity Values for Problem1, $X = W_g = W_p = 0.5$.

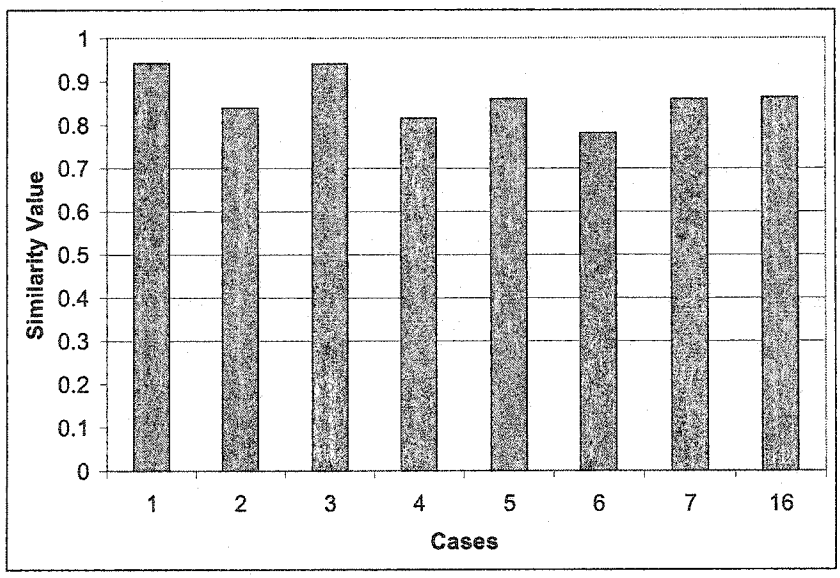


Figure 5.12: Similarity Values for Problem1, $X = 1$, $W_g = W_p = 0.5$.

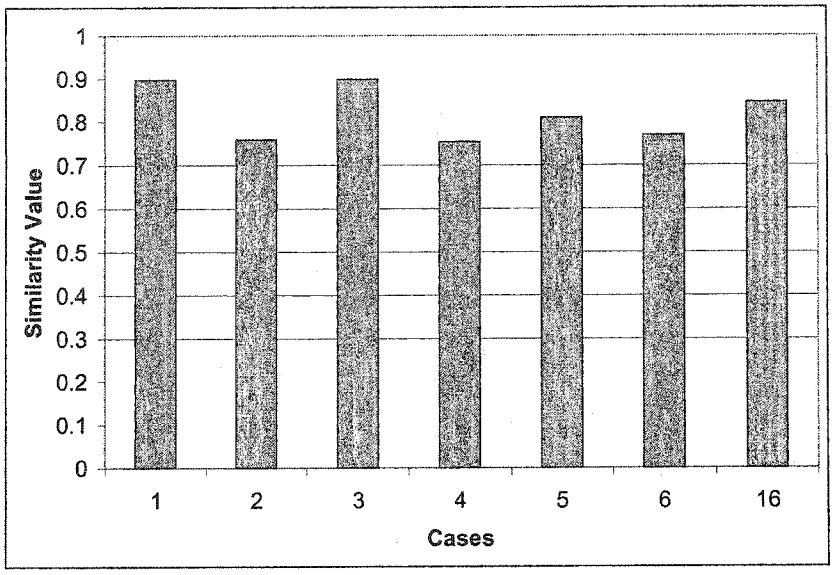


Figure 5.13: Similarity Values for Problem1, $X = 0.8$, $W_g = W_p = 0.5$.

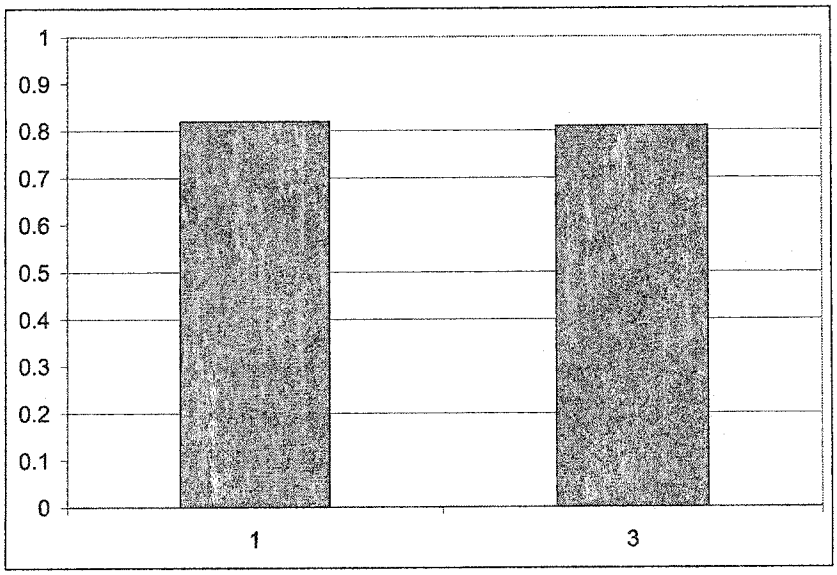


Figure 5.14: Similarity Values for Problem1, $X = 0.3$, $W_g = W_p = 0.5$.

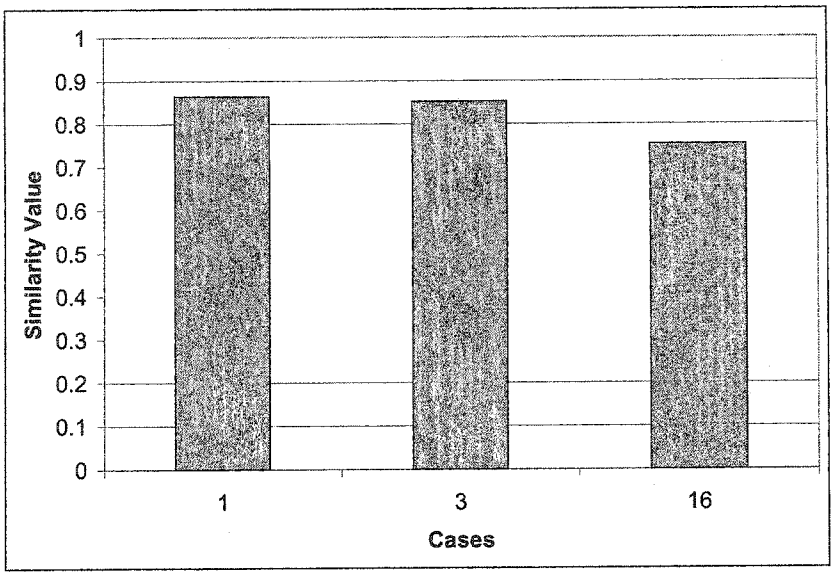


Figure 5.15: Similarity Values for Problem1, $X = 0.5$, $W_g = 0.3$, $W_p = 0.7$.

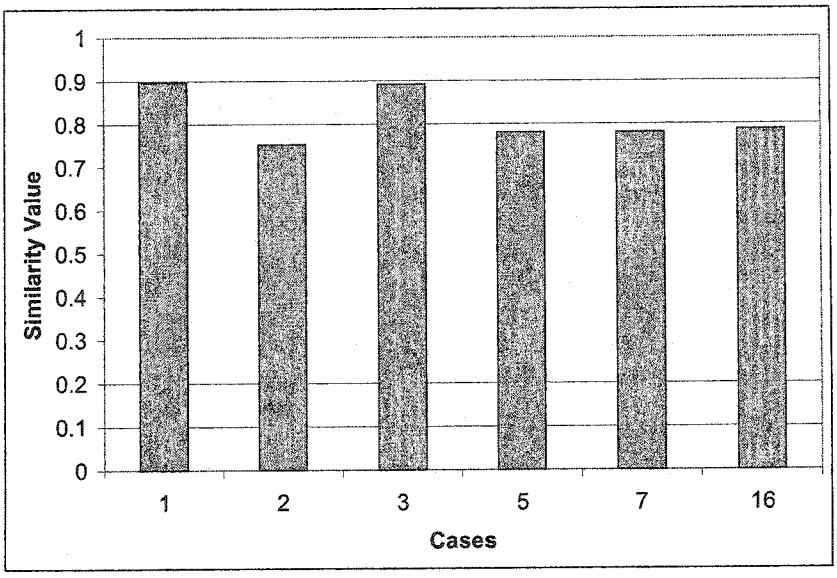


Figure 5.16: Similarity Values for Problem1, $X = 0.8$, $W_g = 0.3$, $W_p = 0.7$.

	Before Replay	After Replay 1	After Replay 2	After Replay 3	After Replay 4
Case#	Sim. Value 1	Sim. Value 2	Sim. Value 3	Sim. Value 4	Sim. Value 5
1	**0.876667**	**0.826294**	**0.796003**	0.741827	x
2	**0.7740480**.6	0829	0.688382	x	x
3	**0.873022**	**0.857348**	**0.857348**	**0.864102**	**0.865827**
4	0.743208	0.647225	0.660725	x	x
5	**0.794494**	0.733634	0.733634	x	x
6	0.733694	x	0.733694	x	x
7	**0.793341**	0.740407	0.728824	x	x
8	0.3	0.3	0.3	x	x
9	0.230769	0.230769	0.230769	x	x
10	0.156596	0.156596	0.156596	x	x
11	0.27832	0.27832	0.27832	x	x
12	0.37498	0.37498	0.37498	x	x
13	0.262342	0.262342	0.262342	x	x
14	0.277211	0.277211	0.277211	x	x
15	0.355852	0.355852	0.355852	x	x
16	**0.799912**	**0.792533**	**0.792531**	**0.805212**	**0.807084**

Figure 5.17: Plan replay effect on Problem1.

	Before Replay	After Replay 1	After Replay 2
Case#	Sim. Value 1	Sim. Value 2	Sim. Value 3
1	0.684524	0.63476	x
2	0.6914636	0.639456	x
3	**0.77945**	0.600792	x
4	0.575354	0.544901	x
5	**0.789466**	**0.767018**	**0.767018**
6	**0.762139**	0.702286	0.674815
7	**0.750078**	**0.773121**	**0.773121**
8	0.214342	0.179831	x
9	0.347826	0.393162	x
10	0.249531	0.24933	x
11	0.24125	0.236938	x
12	0.295769	0.375792	x
13	0.357595	0.365009	x
14	0.183899	0.185247	x
15	0.36832	0.408313	x
16	0.525053	0.47709	x

Figure 5.18: Plan replay effect on Problem2.

	Before Replay	After Replay 1	After Replay 2	After Replay 3	After Replay 4
Case#	Sim. Value 1	Sim. Value 2	Sim. Value 3	Sim. Value 4	Sim. Value 5
1	**0.9166670.9**	**16667**	**0.916667**	**0.916667**	**0.916667**
2	**0.8139210.8**	**19327**	**0.819327**	**0.840506**	**0.840506**
3	**0.7590490.7**	**53877**	**0.753877**	**0.756528**	**0.756528**
4	0.619291	0.524876	0.536204	x	x
5	0.658581	0.607307	0.607307	x	x
6	0.59825	0.59825	x	x	x
7	0.656546	0.611054	x	x	x
8	0.296429	0.296429	x	x	x
9	0.230769	0.230769	x	x	x
10	0.272266	0.334185	x	x	x
11	0.446392	0.446392	x	x	x
12	0.369028	0.369028	x	x	x
13	0.254145	0.254145	x	x	x
14	0.394238	0.394238	x	x	x
15	0.35801	0.35801	x	x	x
16	**0.7756670.8**	**20708**	**0.753719**	0.720324	x

Figure 5.19: Plan replay effect on Problem3.

	Before Replay	After Replay 1	After Replay 2
Case #	Sim. Value 1	Sim. Value 2	Sim. Value 3
1	**0.847375**	0.698838	0.586163
2	**0.939635**	**0.760283**	0.702601
3	0.67275	x	x
4	**0.750695**	0.613032	0.554398
5	0.606655	x	x
6	0.63663	x	x
7	**0.785847**	0.514544	0.500501
8	0.325661	x	x
9	0.418803	x	x
10	0.420056	x	x
11	0.572656	x	x
12	0.500357	x	x
13	0.20745	x	x
14	0.582736	x	x
15	0.293542	x	x
16	**0.81828**	0.643475	0.593128

Figure 5.20: Plan replay effect on Problem4.

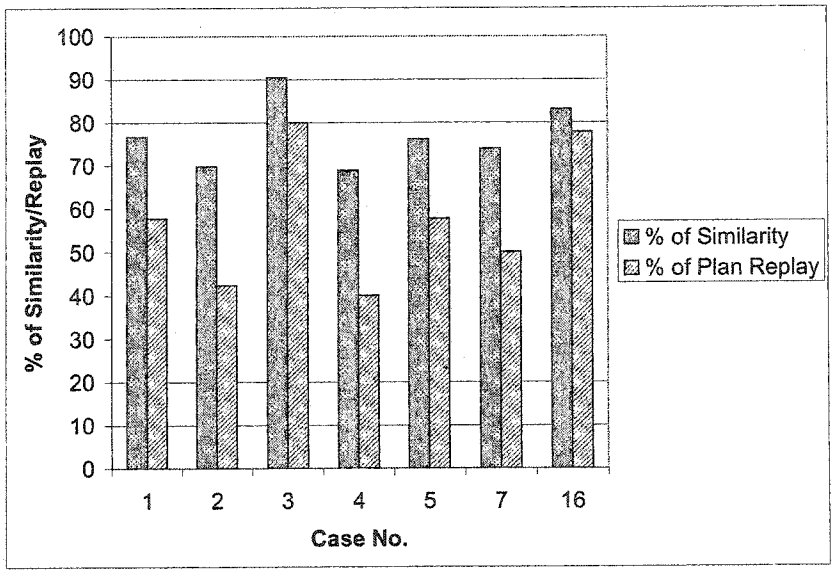


Figure 5.21: Soundness Test 1: % of Similarity Value and Case Plan Replayed

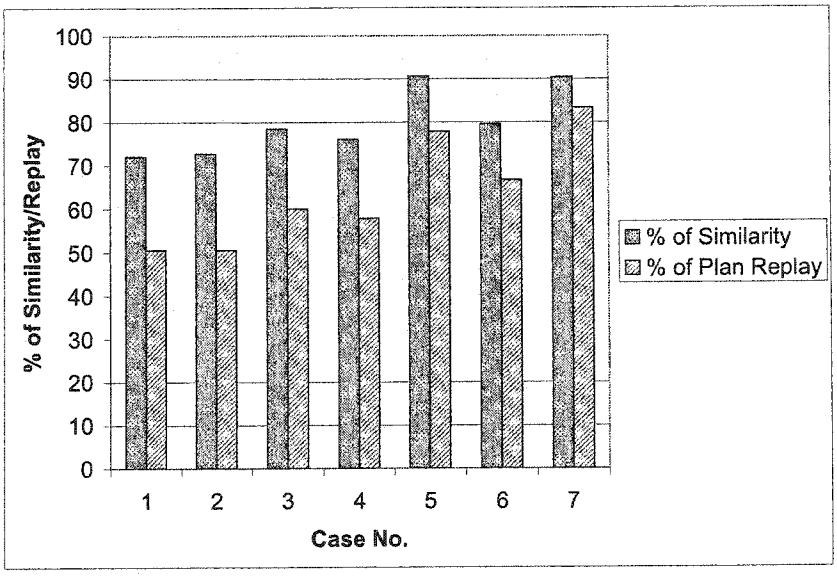


Figure 5.22: Soundness Test 2: % of Similarity Value and Case Plan Replayed

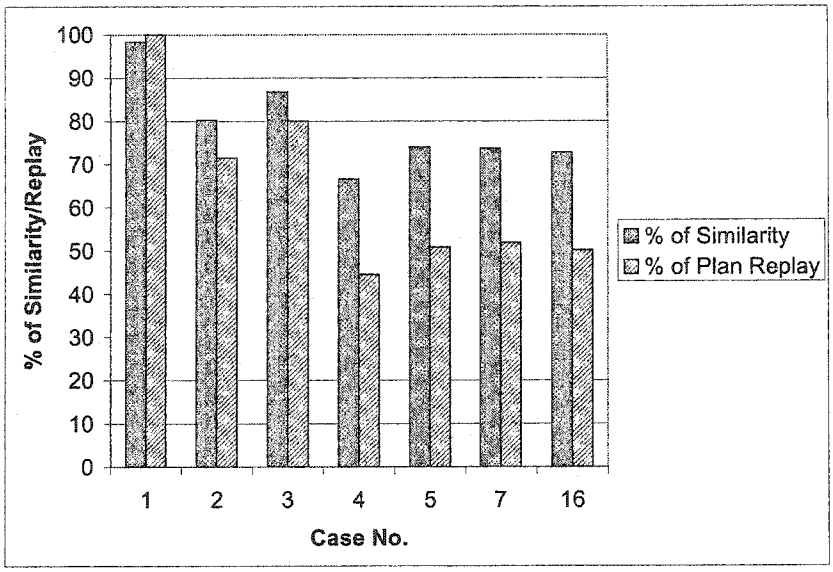


Figure 5.23: Soundness Test 3: % of Similarity Value and Case Plan Replayed

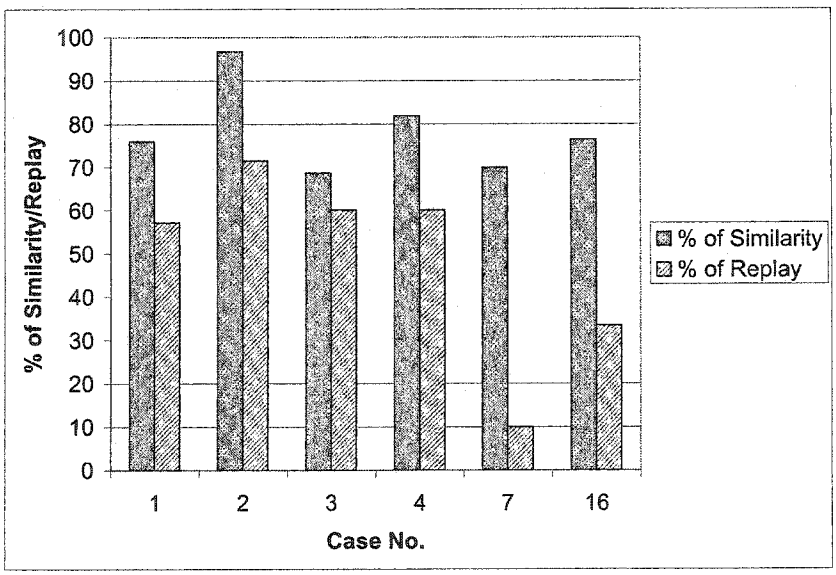


Figure 5.24: Soundness Test 4: % of Similarity Value and Case Plan Replayed

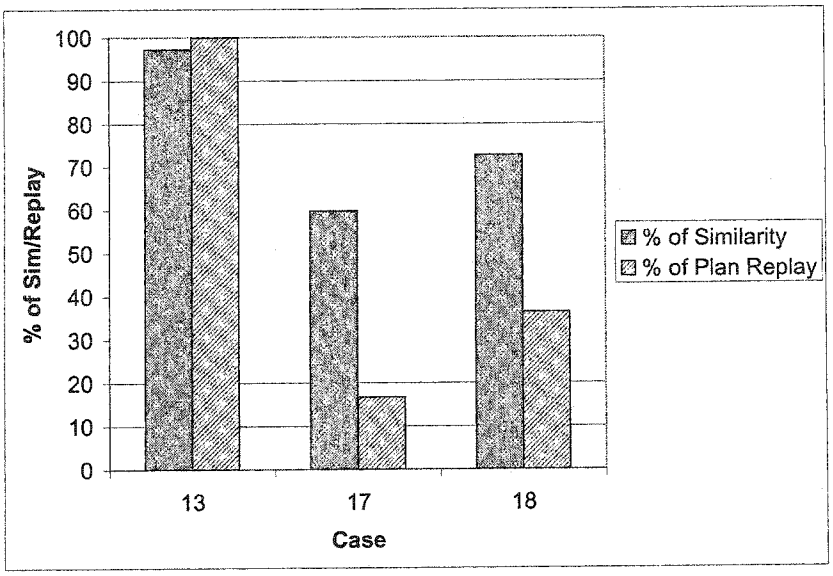


Figure 5.25: Soundness Test 5: % of Similarity Value and Case Plan Replayed

Pb. #	No. of Obj.	No. of Pred.	ES Time (sec)	TS Time (sec)	ES Sim.	TS Sim.
1	5	42	10.875	67.887	0.9167	0.9167
2	6	30	312.1345	90.55	0.7545	0.7545
3	7	56	576.018	174.771	0.8546	0.8546
4	8	68	$6.99E+03$	323.355	0.9479	0.9479
5	9	50	$4.76E+04$	744.251	0.8348	0.8348

Table 5.12: Comparison Table for TS and ES Times

a problem. A comparison between TS and SA best solution times is shown in Figure ??. We used PentiumIII, 700 MHz machine with 256 MB RAM for performing these experiments. This time can be further reduced if we use the parallel support for case-based planning discussed in [88], which shows a drastic improvement in retrieval process of complex cases for very large and unindexed memory.

Pb. #	No. of Obj.	No. of Pred.	ES Time (sec)	TS Time (sec)	ES Sim.	TS Sim.
1	7	58	576.0180	174.7710	0.8546	0.8546
2	7	53	510.6440	159.7800	0.6605	0.6605
3	7	52	485.9490	152.5490	0.7449	0.7449
4	7	56	495.2830	161.8330	0.6984	0.6984

Table 5.13: TS and ES Times for same no. of objects and different no. of predicates

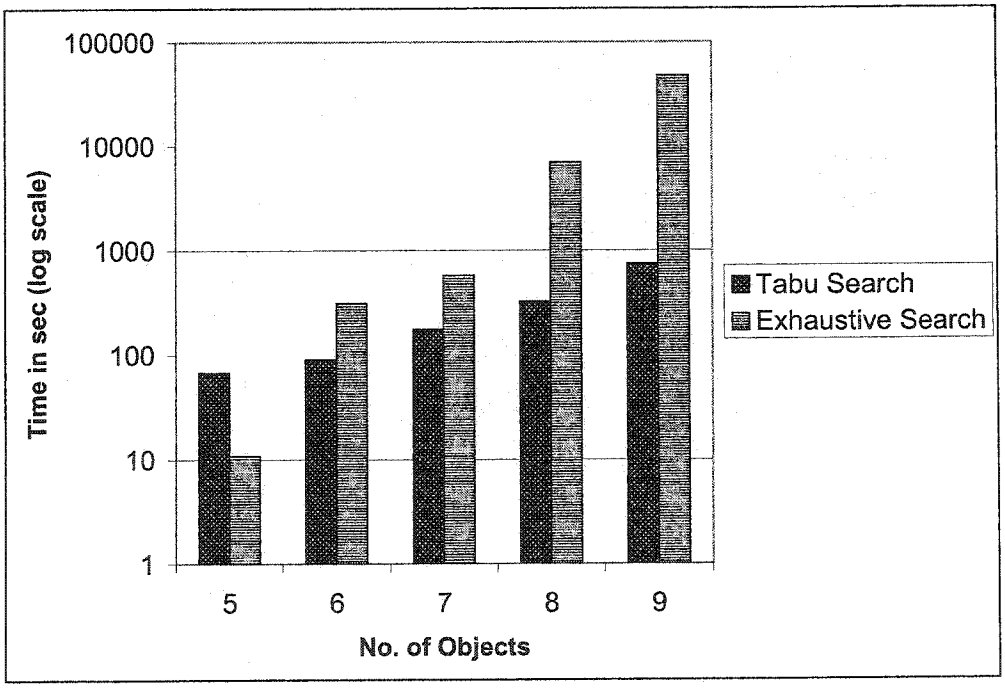


Figure 5.26: Comparison of TS and ES Times

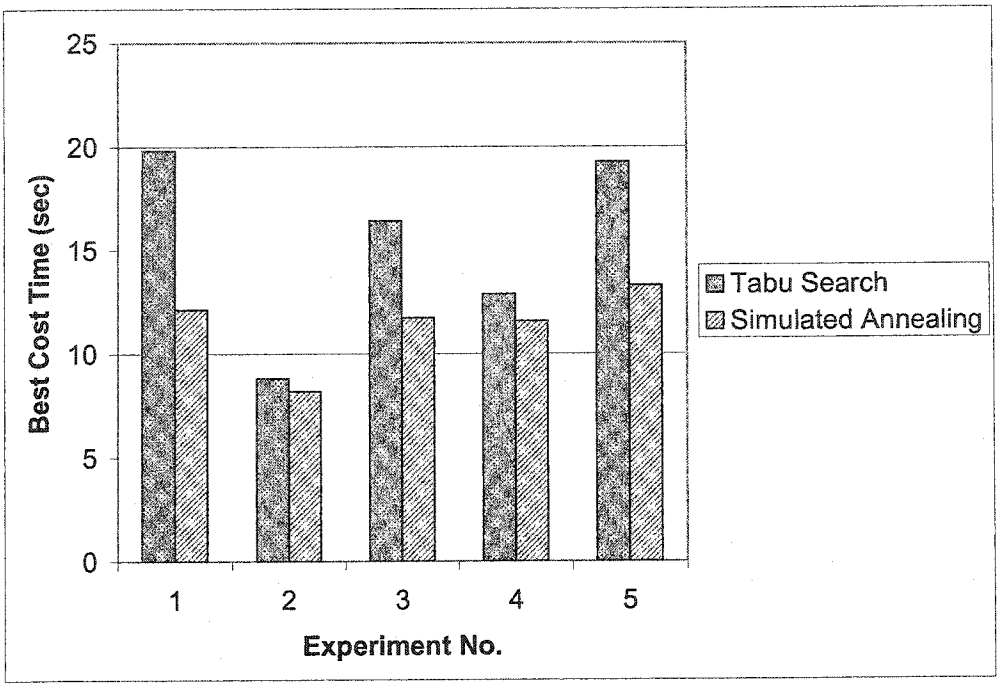


Figure 5.27: Comparison of SA and TS best solution Times

5.6 Conclusion

In this chapter, we presented and discussed various experiments that were performed using our developed similarity metric FDSM. FDSM has performed well for both crisp predicate based domain and fuzzy predicate based domain and it is effective in retrieving the similar cases. Fuzzy plan replay system was developed for similarity assessment and learning of feature weights. For updating the weights of features we used weighting model presented in [83] and [84]. For instantiating the predicates, we compared Exhaustive Search (ES) ,Tabu Search (TS) and Simulated Annealing (SA). We found that SA is more efficient when compared to TS and ES.

Chapter 6

Conclusion and Future Work

Planning has been a subject of considerable research effort in artificial intelligence, due largely to its potential for use in a wide range of practical applications [89]. Planning frequently involves searching a large space of possible actions, thus developing techniques that efficiently guide search in planning is an important research topic. Case-based planning (CBP)systems, an important class of case-based reasoner (CBR), solve this problem by caching solutions to planning problems in case base, and by providing methods for retrieving solutions whose problems are similar to a new, given problem. In this work, we have discussed CBP under imperfect environment. Mainly, we concentrated on the retrieval of similar cases, which is the first and most crucial step in CBP. We conclude this thesis by summarizing our major contributions and discussing future research directions.

6.1 Major Contributions

Our major contributions in this work are described below:

- We have developed a fuzzy predicate based similarity metric for CBP. This similarity metric assesses similarity using a similarity measure (described in chapter 4) between case and new problem predicates. Our developed metric is applicable both in crisp and fuzzy predicate based specified domains.

- Our similarity metric is dynamic i.e. it learns from previous case-base planning episodes. For this purpose, weights of a case predicates are incorporated in the similarity metric. Initially, weight of each footprint predicate is computed statically by observing its effect on all actions in the plan and its occurrence as a precondition for them. The weights of footprint predicates are updated depending on the feedback from repairer; if the retrieval is successful, the weights of certain predicates are incremented by a factor which depends on previous performance of the case. Otherwise, the weights of the features are decremented by the same factor. The weights of new problem goals are also incorporated in our similarity metric. In addition, initial state and goal state (situation) weight factors are also included for adjusting the importance of initial state and goal state (situation) similarity values. These factors may be provided by the user with new problem or they can be learnt by some heuristics.

- For instantiating the predicates appropriately, so that we could get maximum

similarity between a case and a problem, we have used Tabu Search (TS) and Simulate Annealing (SA). We have also compared TS with exhaustive search (ES) and it was found, for problems having large number of objects, the performance of ES is very poor. On the other hand, TS gives satisfactory results in acceptable time. We also compared SA with TS and found that SA performs better than TS in our planning domains.

- We have developed a plan replaying system using fuzzy inference engine for simulation purpose. This system replays retrieved cases (cases having similarity value greater than a certain threshold) and counts the number of plan actions satisfied for the new problem. On the basis of these two (satisfied actions and repairing effort) the weights of case predicates are updated.

- We have implemented similarity metrics presented in [78] (for fuzzy reasoning methods)and [75], and have compared our similarity metric with them. Both of these methods ([78] and [75]) are used for fuzzy rule based systems. We have tried them for CBP with fuzzy predicates and compared their results. Also, we have implemented two similarity metrics based on crisp footprint predicates presented in [18] and [25], and have compared them with FDSM.

6.2 Future directions

Following are some future research directions of our work:

- Analytical validation of fuzzy predicate based similarity metric (FDSM).
- For instantiating solution parameters parallel Tabu Search/Simulate Annealing can be used, which will further reduce the retrieval time.
- For making whole process of case-based planning more efficient and effective, plan representation and memory organization needs further exploration.
- Graphical user interface for retrieving similar cases from memory.

Appendix A

The Logistic Transportation Domain

Taken from [19]

Object types: (8)

Airplane
Superclass: Carrier
Airport
Superclass: Location
Carrier
Superclass: None.
City
Superclass: None.
Location
Superclass: None.
Object
Superclass: None.
PostOffice
Superclass: Location
Truck
Superclass: Carrier

Predicates: (9)

atAirplane
Arguments: airplane loc
atObj
Arguments: obj loc
atTruck
Arguments: truck loc
diffCity
Arguments: loc1 loc2
insideAirplane
Arguments: obj plane
insideTruck
Arguments: obj airplane
locationAt
Arguments: loc city
sameCity
Arguments: loc1 loc2
truckInCity
Arguments: truck city

Operators: (6)

DriveTruck
Arguments: truck locFrom locTo city
Constraints:
IsOfType(Truck, truck)
IsOfType(Location, locFrom, locTo)
IsOfType(City, city)
NotSame(locFrom, locTo)
Purposes:
+atTruck(truck, locTo)
atTruck(truck, locFrom)
Preconditions:
+atTruck(truck, locFrom)
+locationAt(locFrom, city)
+sameCity(locFrom, locTo)
+truckInCity(truck, city)
+locationAt(locTo, city)

FlyAirplane

Arguments: airplane locFrom locTo
Constraints:
IsOfType(Airplane, airplane)
IsOfType(Airport, locFrom, locTo)
NotSame(locFrom, locTo)
Purposes: +atAirplane(airplane, locTo)
atAirplane(airplane, locFrom)
Preconditions: +atAirplane(airplane, locFrom)
+diffCity(locFrom, locTo)

LoadAirplane
Arguments: obj airplane loc
Constraints:
IsOfType(Object, obj)
IsOfType(Airplane, airplane)
IsOfType(Airport, loc)
Purposes: +insideAirplane(obj, airplane)
atObj(obj, loc)
Preconditions:
+atObj(obj, loc)
+atAirplane(airplane, loc)

LoadTruck
Arguments: obj truck loc
Constraints:
IsOfType(Object, obj)
IsOfType(Truck, truck)
IsOfType(Location, loc)
Purposes:
+insideTruck(obj, truck)
atObj(obj, loc)
Preconditions:
+ atObj(obj, loc)
+atTruck(truck, loc)
UnloadAirplane
Arguments: obj airplane loc
Constraints:
IsOfType(Object, obj)
IsOfType(Airplane, airplane)
IsOfType(Airport, loc)
Purposes:
+atObj(obj, loc)

insideAirplane(obj, airplane)
Preconditions:
+insideAirplane(obj, airplane)
+atAirplane(airplane, loc)
UnloadTruck
Arguments: obj truck loc
Constraints:
IsOfType(Object, obj)
IsOfType(Truck, truck)
IsOfType(Location, loc)
Purposes:
+atObj(obj, loc)
insideTruck(obj, truck)
Preconditions:
+insideTruck(obj, truck)
+atTruck(truck, loc)

Appendix B

The Battle Field Domain

Object types: ()
Name: Plane
SuperClass: None

Name: Unit
SuperClass: None

Name: Area
SuperClass: None

Name: Weapon
SuperClass: None

Name: Vehicle
SuperClass: None

Name: Power
SuperClass: None

Name: Bridge
SuperClass: None

Name: Missile
SuperClass: Weapon

Name: Gun
SuperClass: Weapon

Name: Ground Transport
SuperClass: Transport

Name: Bomber
SuperClass: Plane

Name: Fighter
SuperClass: Plane

Name: Cargo
SuperClass: Plane

Name: Supporter
SuperClass: Plane

Name: Spy
SuperClass: Plane

Name: Location
SuperClass: Area

Predicates: ()

Name: Plane_NearAt_Location
Abbrv: PNL
Args: P:Plane, L:Location
Type: F
Range: 5:10:20

Name: Plane_VeryNearAt_Location
Abbrv: PVNL
Args: P:Plane, L:Location
Type: F
Range: 1:5:10

Name: Plane_FarFrom_Location
Abbrv: PFL
Args: P:Plane, L:Location
Type: F
Range: 20:40:60

Name: Plane_VeryFarFrom_Location
Abbrv: PVFL
Args: P:Plane, L:Location
Type: F
Range: 55:95:100

Name: Unit_NearAt_Location
Abbrv: UNL
Args: U:Unit, L:Location
Type: F
Range: 5:10:20

Name: Unit_VeryNearAt_Location
Abbrv: UVNL
Args: U:Unit, L:Location
Type: F
Range: 1:5:10

Name: Unit_FarFrom_Location
Abbrv: UFL
Args: U:Unit, L:Location
Type: F
Range: 20:40:60

Name: Unit_VeryFarFrom_Location
Abbrv: UVFL
Args: U:Unit, L:Location
Type: F
Range: 55:95:100

Name: Missile_At_Location
Abbrv: MAL Args: M:Missile, L:Location
Type: NF

Name: Power_Station_At_Location
Abbrv: WAL
Args: W:Power Station, L:Location
Type: NF

Name: Bridge_At_Location
Abbrv: BAL
Args: B:Bridge, L:Location
Type: NF

Name: Gun_At_Location
Abbrv: GAL
Args: G:Gun, L:Location
Type: NF

Name: Location_Near_At_Location
Abbrv: LNL
Args: L:Location, L:Location
Type: F
Range: 5:10:20

Name: Location_VeryNearAt_Location
Abbrv: LVNL
Args: L:Location, L:Location
Type: F
Range: 1:5:10

Name: Location_FarFrom_Location
Abbrv: LFL
Args: L:Location, L:Location
Type: F
Range: 20:40:60

Name: Location_VeryFarFrom_Location
Abbrv: LVFL
Args: L:Location, L:Location
Type: F
Range: 55:95:100

Name: Unit_Available
Abbrv: UA
Args: U:Unit
Type: NF

Name: Plane_Available
Abbrv: PA
Args: P:Plane
Type: NF

Name: Missile_Available
Abbrv: MA
Args: M:Missile
Type: NF

Name: Ground_Transport_Available
Abbrv: GTA
Args: R:Transport
Type: NF

Name: Gun_Available
Abbrv: GA
Args: G:Gun
Type: NF

Name: Low_Casualities
Abbrv: LC
Args:
Type: F
Range: 200:300:400

Name: Very_High_Casualities
Abbrv: VHC
Args:
Type: F
Range: 2000:6000:8000

Name: High_Casualities
Abbrv: HC
Args:
Type: F
Range: 500:1000:2000

Name: Very_Low_Casualities
Abbrv: VLC
Args:
Type: F
Range: 10:50:100

Name: Medium_Casualities
Abbrv: MC
Args:
Type: F
Range: 350:500:550

Name: VeryHigh_Cost_Plane
Abbrv: VHCP
Args: P:Plane
Type: F
Range: 8:10:10

Name: High_Cost_Plane
Abbrv: HCP
Args: P:Plane
Type: F
Range: 7:8:9

Name: Medium_Cost_Plane
Abbrv: MCP
Args: P:Plane
Type: F
Range: 6:7:8

Name: Low_Cost_Plane
Abbrv: LCP
Args: P:Plane
Type: F
Range: 5:6:7

Name: VeryHigh_Cost_Missile
Abbrv: VHCM
Args: M:Missile
Type: F
Range: 8:10:10

Name: High_Cost_Missile
Abbrv: HCM
Args: M:Missile
Type: F
Range: 7:8:9

Name: Low_Cost_Missile
Abbrv: LCM
Args: M:Missile
Type: F
Range: 5:6:7

Name: Medium_Cost_Missile
Abbrv: MCM
Args: M:Missile
Type: F
Range: 6:7:8

Name: WAR_Condition
Abbrv: WC
Args:
Type: NF

Name: Peace_Condition
Abbrv: PC
Args:
Type: NF

Name: Unit_Comp_Destroyed
Abbrv: UCD
Args: U:Unit
Type: F
Range: 95:100:100

Name: Unit_Partially_Destroyed
Abbrv: UPD
Args: U:Unit
Type: F
Range: 50:65:75

Name: Unit_MoreOrLess_Destroyed
Abbrv: UMD
Args: U:Unit
Type: F
Range: 70:85:95

Name: Unit_Little_Destroyed
Abbrv: ULD

Args: U:Unit
Type: F
Range: 10:25:30

Name: Plane_Comp_Destroyed
Abbrv: PCD
Args: P:Plane
Type: F
Range: 95:100:100

Name: Plane_Partially_Destroyed
Abbrv: PPD
Args: P:Plane
Type: F
Range: 50:65:75

Name: PowerUnit_Comp_Destroyed
Abbrv: WCD
Args: W:Power Unit
Type: F
Range: 95:100:100

Name: PowerUnit_Partially_Destroyed
Abbrv: WPD
Args: W:Power Unit
Type: F
Range: 50:65:75

Name: Bridge_Comp_Destroyed
Abbrv: BCD
Args: B:Bridge
Type: F
Range: 95:100:100

Name: Bridge_Partially_Destroyed
Abbrv: BPD
Args: B:Bridge
Type: F
Range: 50:65:75

Name: Enemy_Plane
Abbrv: EP
Args: P:Plane
Type: NF

Name: Enemy_Unit
Abbrv: EU
Args: U:Unit
Type: NF

Name: Enemy_Missile
Abbrv: EM
Args: M:Missile
Type: NF

Name: Enemy_Location
Abbrv: EL
Args: L:Location
Type: NF

Name: Enemy_Power_Station
Abbrv: EW
Args: W:Power
Type: NF

Name: Enemy_Bridge
Abbrv: EB
Args: B:Bridge
Type: NF

Name: IS_A_Plane_Bomber
Abbrv: ISAPB
Args: P:Plane
Type: NF

Name: IS_A_Plane_Fighter
Abbrv: ISAPF
Args: P:Plane
Type: NF

Name: IS_A_Plane_Cargo
Abbrv: ISAPC
Args: P:Plane
Type: NF

Name: IS_A_Plane_Support
Abbrv: ISAPS
Args: P:Plane
Type: NF

Name: IS_A_Plane_Spy
Abbrv: ISAPY
Args: P:Plane
Type: NF

Name: IS_A_Unit
Abbrv: ISAU
Args: U:Unit
Type: NF

Name: IS_A_Missile
Abbrv: ISAM
Args: M:Missile
Type: NF

Name: IS_A_Neuclear_Missile
Abbrv: ISANM
Args: M:Missile
Type: NF

Name: IS_A_Ground_Transport
Abbrv: ISAGT
Args: R:Ground_Transport
Type: NF

Name: IS_A_Power_Station
Abbrv: ISAWS
Args: W:Power_Station
Type: NF

Name: IS_A_Bridge
Abbrv: ISAB
Args: B:Bridge
Type: NF

Name: IS_A_Location
Abbrv: ISAL
Args: L:Location
Type: NF

Name: Occupied_Area_Comp
Abbrv: OAC
Args: A:Area
Type: F
Range: 95:100:100

Name: Occupied_Area_Partially
Abbrv: OAP
Args: A:Area
Type: F
Range: 50:65:75

Name: occupied_Area_little
Abbrv: OAL
Args: A:Area
Type: F
Range: 10:25:30

Name: occupied_Area_MoreOrLess

Abbrv: OAM
Args: A:Area
Type: F
Range: 70:85:95

Name: Positioned_Unit_Near_Location
Abbrv: PNL
Args: U:Unit, L:Location
Type: F
Range: 10:20:30

Name: Positioned_Unit_VeryNear_Location
Abbrv: PVL
Args: U:Unit, L:Location
Type: F
Range: 0:5:10

Name: Positioned_Unit_FarFrom_Location
Abbrv: PFL
Args: U:Unit,L:Location
Type: F
Range: 30:50:70

Name: Positioned_Unit_Near_Unit
Abbrv: PNU
Args: U:Unit, U:Unit
Type: F
Range: 10:20:30

Name: Positioned_Unit_VeryNear_Unit
Abbrv: PVU
Args: U:Unit, U:Unit
Type: F
Range: 0:5:10

Name: Positioned_Unit_FarFrom_Unit
Abbrv: PFU
Args: U:Unit, U:Unit
Type: F
Range: 30:50:70

Name: Within_Plane_Range_Plane
Abbrv: WPRP
Args: P:Plane, P:Plane
Type: F
Range: 5:10:20

Name: OutOf_Plane_Range_Plane
Abbrv: OPRP

Args: P:Plane, P:Plane
Type: F
Range: 180:190:200

Name: Within_Plane_Min_Range_Plane
Abbrv: WPMRP
Args: P:Plane, P:Plane
Type: F
Range: 10:20:25

Name: Within_Plane_Range_Unit
Abbrv: WPRU
Args: P:Plane, U:Unit
Type: F
Range: 5:10:20

Name: OutOf_Plane_Range_Unit
Abbrv: OPRU
Args: P:Plane, U:Unit
Type: F
Range: 180:190:200

Name: Within_Plane_Min_Range_Unit
Abbrv: WPMRU
Args: P:Plane, U:Unit
Type: F
Range: 10:20:25

Name: Within_Plane_Range_Location
Abbrv: WPRL
Args: P:Plane, L:Location
Type: F
Range: 5:10:20

Name: OutOf_Plane_Range_Location
Abbrv: OPRL
Args: P:Plane, L:Locatoin
Type: F
Range: 180:190:200

Name: Within_Plane_Min_Range_Location
Abbrv: WPMRL
Args: P:Plane, L:Location
Type: F
Range: 10:20:25

Name: Within_Missile_Range_Plane
Abbrv: WMRP

Args: M:Missile, P:Plane
Type: F
Range: 5:10:20

Name: OutOf_Missile_Range_Plane
Abbrv: OMRP
Args: M:Missile, P:Plane
Type: F
Range: 180:190:200

Name: Within_Missile_Min_Range_Plane
Abbrv: WMMRP
Args: M:Missile, P:Plane
Type: F
Range: 10:20:25

Name: Within_Missile_Range_Location
Abbrv: WMRL
Args: M:Missile, L:Location
Type: F
Range: 5:10:20

Name: OutOf_Missile_Range_Location
Abbrv: OMRL
Args: M:Missile, L:Locatoin
Type: F
Range: 180:190:200

Name: Within_Missile_Min_Range_Location
Abbrv: WMMRL
Args: M:Missile, L:Location
Type: F
Range: 10:20:25

Name: Within_Gun_Range_Plane
Abbrv: WGRP
Args: G:Gun, P:Plane
Type: F
Range: 5:10:20

Name: OutOf_Gun_Range_Plane
Abbrv: OGRP
Args: G:Gun, P:Plane
Type: F
Range: 180:190:200

Name: Within_Gun_Min_Range_Plane
Abbrv: WGMRP

Args: G:Gun, P:Plane
Type: F
Range: 10:20:25

Name: Unit_Ready
Abbrv: UR
Args: U:Unit
Type: NF

Name: Plane_Ready
Abbrv: PR
Args: P:Plane
Type: NF

Name: Missile_Ready
Abbrv: MR
Args: M:Missile
Type: NF

Name: Ground_Transport_Ready
Abbrv: GTR
Args: R:Ground_Transport
Type: NF

Name: Gun_Ready
Abbrv: GR
Args: G:Gun
Type: NF

Name: Seize_Area_Comp
Abbrv: SAC
Args: A:Area
Type: F
Range: 95:100:100

Name: Seize_Area_Partially
Abbrv: SAP
Args: A:Area
Type: F
Range: 50:65:75

Name: Seize_Area_MoreOrLess
Abbrv: SAM
Args: A:Area
Type: F
Range: 70:85:95

Name: shifted_Unit_Comp
Abbrv: HUC

Args: U:Unit
Type: F
Range: 95:100:100

Name: shifted_Unit_Partially
Abbrv: HUP
Args: U:Unit
Type: F
Range: 50:65:75

Name: shifted_Unit_Little
Abbrv: HUL
Args: U:Unit
Type: F
Range: 10:25:30

Name: shifted_Unit_MoreOrLess
Abbrv: HUM
Args: U:Unit
Type: F
Range: 70:85:95

Name: VeryHigh_Speed_Plane
Abbrv: VHSP
Args: P:Plane
Type: F
Range: 420:540:540

Name: High_Speed_Plane
Abbrv: HSP
Args: P:Plane
Type: F
Range: 340:410:425

Name: Medium_Speed_Plane
Abbrv: MSP
Args: P:Plane
Type: F
Range: 300:320:360

Name: Low_Speed_Plane
Abbrv: LSP
Args: P:Plane
Type: F
Range: 200:275:300

Name: VeryHigh_Speed_Missile
Abbrv: VHSM

Args: M:Missile
Type: F
Range: 420:540:540

Name: High_Speed_Missile
Abbrv: HSM
Args: M:Missile
Type: F
Range: 340:410:425

Name: Medium_Speed_Missile
Abbrv: MSM
Args: M:Missile
Type: F
Range: 300:320:360

Name: LowSpeed_Missile
Abbrv: LSM
Args: M:Missile
Type: F
Range: 200:275:300

Name: VeryHigh_Speed_Ground_Transport
Abbrv: VSR
Args: R:Ground_Transport
Type: F
Range: 200:275:300

Name: High_Speed_Ground_Transport
Abbrv: HSR
Args: R:Ground_Transport
Type: F
Range: 150:175:210

Name: Medium_Speed_Ground_Transport
Abbrv: MSR
Args: R:Ground_Transport
Type: F
Range: 100:140:160

Name: Low_Speed_Ground_Transport
Abbrv: LSR
Args: R:Ground_Transport
Type: F
Range: 75:90:110

Name: VeryHilly_Terrain_AtLocatoin
Abbrv: VHTL

Args: L:Location
Type: F
Range: 85:100:100

Name: Hilly_Terrain_AtLocatoin
Abbrv: HTL
Args: L:Location
Type: F
Range: 55:80:90

Name: LessHilly_Terrain_AtLocatoin
Abbrv: LHTL
Args: L:Location
Type: F
Range: 15:35:55

Name: NotHilly_Terrain_AtLocatoin
Abbrv: NHTL
Args: L:Location
Type: F
Range: 0:10:15

Name: VeryHigh_Threat_AtLocation
Abbrv: VTHL
Args: L:Location
Type: F
Range: 85:100:100

Name: High_Threat_AtLocation
Abbrv: HTHL
Args: L:Location
Type: F
Range: 70:80:90

Name: Low_Threat_AtLocation
Abbrv: LTHL
Args: L:Location
Type: F
Range: 10:35:45

Name: Medium_Threat_AtLocation
Abbrv: MTHL
Args: L:Location
Type: F
Range: 40:65:70

Name: No_Threat_AtLocation
Abbrv: NTHL

Args: L:Location
Type: F
Range: 0:1:1

Name: Fine_Weather_AtLocatoin
Abbrv: FWL
Args: L:Location
Type: F
Range: 50:65:80

Name: VeryFine_Weather_AtLocatoin
Abbrv: VFWL
Args: L:Location
Type: F
Range: 75:85:100

Name: NotFine_Weather_AtLocatoin
Abbrv: NFWL
Args: L:Location
Type: F
Range: 10:25:50

Operators: ()

FPlaneGun_FireAt_Unit (FPGFU)
Args: Plane Unit Loc
Precond:
ISAPF(P1)
ISAU(U1)
ISALOC(L1)
ISAG(G1)
PHAVEG(P1,G1)
UNL(U1,L1)
EU(U1)
WGRU(G1,U1,L1)
PNL(P1,L1)
Post:
+ UCD(U1)
−UNL(U1,L1)

FPlaneGun_FireAt_FPlane (FPGFP)
Args: Plane Plane Loc
Precond:
ISAPF(P1)

ISAFP(P2)
ISALOC(L1)
ISAG(G1)
PHAVEG(P1,G1)
PVNL(P2,L1)
EP(P2)
WGRP(G1,P2,L1)
PNL(P1,L1)
Post: + PCD(P2)
−PVNL(P2,L1)

FPlaneMissile_FireAt_Unit (FPMFU)
Args: Plane Unit Loc
Precond:
ISAPF(P1)
ISAU(U1)
ISALOC(L1)
ISAM(M1)
PHAVEM(P1,M1)
UNL(U1,L1)
EU(U1)
WMRU(M1,U1,L1)
PNL(P1,L1)
Post: + UCD(U1)
−UNL(U1,L1)

FPlaneMissile_FireAt_FPlane (FPMFP)
Args: Plane Plane Loc
Precond:
ISAPF(P1)
ISAPF(P2)
ISALOC(L1)
ISAM(M1)
PHAVEM(P1,M1)
PVNL(P2,L1)
EP(P2)
WMRP(M1,P2,L1)
PNL(P1,L1)
Post: +PCD(P2)
−PVNL(P2,L1)

UnitGun_FireAt_Unit (UGFU)
Args: Unit1 Unit2 Loc1 Loc2

Precond:
ISAU(U1)
ISAU(U2)
ISALOC(L1)
ISALOC(L2)
ISAG(G1)
UHAVEG(U1,G1)
GAL(G1,L1)
UNL(U1,L1)
EU(U2)
WGRU(G1,U2,L2)
UVNL(U2,L2)
Post: +UCD(U2)
−UVNL(U2,L1)

UnitMissile_FireAt_Unit (UMFU)
Args: Unit1 Unit2 Loc1 Loc2
Precond:
ISAU(U1)
ISAU(U2)
ISALOC(L1)
ISALOC(L2)
ISAM(M1)
UHAVEM(U1,M1)
MNL(M1,L1)
UNL(U1,L1)
EU(U2)
WMRU(M1,U2,L2)
UVNL(U2,L2)
Post: +UCD(U2)
−UVNL(U2,L1)

Fly_Fighter_Plane (FFP)
Args: Plane Loc1 Loc2
Precond:
ISAPF(P1)
ISALOC(L1)
ISALOC(L2)
NOTSAME(L1,L2)
PA(P1)
PR(P1)
PVNL(P1,L1)

Post: +PNL(P1,L2)
−PVNL(P1,L1)

Fly_Spy_Plane (FSP)
Args: Plane Loc1 Loc2
Precond:
ISAPY(P1)
ISALOC(L1)
ISALOC(L2)
NOTSAME(L1,L2)
PA(P1)
PR(P1)
PVNL(P1,L1)
Post: +PNL(P1,L2)
−PVNL(P1,L1)

Move_Unit (MOU)
Args: Unit Loc1 Loc2
Precond:
ISAU(U1)
ISALOC(L1)
ISALOC(L2)
NOTSAME(L1,L2)
UA(U1)
UR(U1)
UVNL(U1,L1)
Post: +UNL(U1,L2)
−UVNL(U1,L1)

Position_Unit (POU)
Args: Unit Loc1
Precond:
ISAU(U1)
ISALOC(L1)
UA(U1)
UR(U1)
UVNL(U1,L1)
Post: +PVL(U1,L1)

Move_Missile (MM)
Args: Missile Loc1 Loc2
Precond:
ISAM(M1)

ISALOC(L1)
ISALOC(L2)
NOTSAME(L1,L2)
MA(M1)
MVNL(M1,L1)
Post: +MNL(M1,L2)
−MVNL(M1,L1)

Set_Gun_AtTarget (SGT)
Args: Gun Loc
Precond:
ISAG(G1)
ISAU(U1)
ISALOC(L1)
GNL(G1,L1)
GA(G1)
GR(G1)
Post: +WGRU(G1,U1,L1)

Set_Missile_AtTarget (SMT)
Args: Missile Loc
Precond:
ISAM(M1)
ISAU(U1)
ISALOC(L1)
MNL(M1,L1)
MA(M1)
MR(M1)
Post: +WMRU(M1,U1,L1)

Set_Camera_AtTarget (SCT)
Args: Camera Loc
Precond:
ISAC(C1)
ISALOC(L1)
CA(C1)
CR(C1)
Post: +WCRL(C1,L1)

SpyPlane_Get_Snaps (SGS)
Args: Plane Loc
Precond:

ISAPY(P1)
ISALOC(L1)
ISASN(SN1)
ISAC(C1)
CR(C1)
PNL(P1,L1)
PHAVEC(P1,C1)
WCRL(C1,L1)
Post: +PHAVESN(P1,SN1)

SpyPlane_Deliver_Snaps_toUnit_ofLoc (SDSU)
Args: Plane Unit Loc
Precond:
ISAPY(P1)
ISALOC(L1)
ISALOC(L2)
ISASN(SN1)
ISAU(U1)
UNL(U1,L2)
PHAVESN(P1,SN1)
Post: −PHAVESN(P1,SN)
+UHAVESN(U1,SN1)

Unit_Occupy_Location (UOL)
Args: Unit Loc
Precond:
ISAU(U1)
ISALOC(L1)
UVNL(U1,L1)
NOENEMY(L1)
Post: +LOCOCC(U1,L1)

Unit_Remove_Enemy_fromLoc (URUL)
Args: Unit1 Unit2 Loc
Precond:
ISAU(U1)
ISAU(U2)
ISALOC(L1)
EU(U2)
UNL(U1,L1)
UVNL(U2,L1)
Post: +NOENEMY(L1)

−UVNL(U2,L1)

Load_Fplane_Gun (LFPG)
Args: Plane Gun
Precond:
ISAPF(P1)
ISAG(G1)
ISAGR(GR1)
GAGR(G1,GR1)
Post: +PHAVEG(P1,G1)
−GAGR(G1,GR1)

Load_Fplane_Missile (LFPM)
Args: Plane Missile
Precond:
ISAPF(P1)
ISAM(M1)
ISAGR(GR1)
MAGR(M1,GR1)
Post: +PHAVEM(P1,M1)
−MAGR(M1,GR1)

Move_Gun_Locfrom_Locto (MGLL)
Args: Gun Loc1 Loc2
Precond:
ISAG(G1)
ISALOC(L1)
ISALOC(L2)
GAL(G1,L1)
Post: +GAL(G1,L2)
−GAL(G1,L1)

Unit_Seize_Area_Completely (USA)
Args: Unit Area Loc
Precond:
ISAU(U1)
ISAA(A1)
ISALOC(L1)
ISINA(L1,A1)
NOENEMY(L1)
UVNL(U1,L1)

Post: +USAC(U1,A1,L1)

Appendix C

Problems

ipname	noargs	arg1	type1	arg2	type2	arg3	type3	ftype	mu
UA	1	U1	UNIT					NF	1
UR	1	U1	UNIT					NF	1
NOTSAME	2	L1	LOCATION	L2	LOCATION			NF	1
MCP	1	P1	PLANE					F	1
VHSP	1	P1	PLANE					F	0.428571
FWL	1	L1	LOCATION					F	0.8
FWL	1	L2	LOCATION					F	0.8
ISAU	1	U1	UNIT					NF	1
ISAU	1	U2	UNIT					NF	1
ISALOC	1	L1	LOCATION					NF	1
ISALOC	1	L2	LOCATION					NF	1
ISALOC	1	L3	LOCATION					NF	1
ISAPB	1	P1	PLANE					NF	1
ISAG	1	G1	GUN					NF	1
NOTSAME	2	L2	LOCATION	L3	LOCATION			NF	1
PA	1	P1	PLANE					NF	1
PR	1	P1	PLANE					NF	1
FWL	1	L3	LOCATION					F	0.8
LHTL	1	L2	LOCATION					F	0.712
PVNL	2	P1	PLANE	L3	LOCATION			F	0.78
GA	1	G1	GUN					NF	1
GR	1	G1	GUN					NF	1
PHAVEG	2	P1	PLANE	G1	GUN			NF	1
UVNL	2	U2	UNIT	L2	LOCATION			F	0.8875
EU	1	U2	UNIT					NF	1
UVNL	2	U1	UNIT	L1	LOCATION			F	0.666667

Figure C.1: Problem 1 Initial State

gpname	nogrgs	arg1	type1	arg2	type2	arg3	type3	ftype	mu	wt
LOCOCC	2	U1	UNIT	L2	LOCATION			F	0.666667	1
UCD	1	U2	UNIT					F	0.666667	1

Figure C.2: Problem 1 Goal State

upname	noargs	arg1	type1	arg2	type2	arg3	type3	ftype	mu
ISALOC	1	L5	LOCATION					NF	1
UVNL	2	U4	UNIT	L4	LOCATION			F	0.8
EU	1	U4	UNIT					NF	1
ISAPF	1	P1	PLANE					NF	1
PHAVEG	2	P1	PLANE	G2	GUN			NF	1
ISAG	1	G2	GUN					NF	1
GA	1	G2	GUN					NF	1
GR	1	G2	GUN					NF	1
PHAVEM	2	P1	PLANE	M1	MISSILE			NF	1
VHSP	1	P1	PLANE					F	0.428571
ISAU	1	U1	UNIT					NF	1
ISAU	1	U2	UNIT					NF	1
ISALOC	1	L1	LOCATION					NF	1
ISALOC	1	L2	LOCATION					NF	1
ISAM	1	M1	MISSILE					NF	1
MA	1	M1	MISSILE					NF	1
MR	1	M1	MISSILE					NF	1
UVNL	2	U1	UNIT	L1	LOCATION			F	0.78
UVNL	2	U2	UNIT	L2	LOCATION			F	0.78
EU	1	U2	UNIT					NF	1
NOTSAME	2	L1	LOCATION	L2	LOCATION			NF	1
UA	1	U1	UNIT					NF	1
UR	1	U1	UNIT					NF	1
UHAVEG	2	U1	UNIT	G1	GUN			NF	1
FWL	1	L1	LOCATION					F	1
FWL	1	L2	LOCATION					F	0.746667
HTL	1	L2	LOCATION					F	0.58
MCM	1	M1	MISSILE					F	1
HSM	1	M1	MISSILE					F	0.874286
ISALOC	1	L3	LOCATION					NF	1
FWL	1	L3	LOCATION					F	0.8
UVNL	2	U3	UNIT	L3	LOCATION			F	0.688889
EU	1	U3	UNIT					NF	1
NOTSAME	2	L1	LOCATION	L3	LOCATION			NF	1
NOTSAME	2	L2	LOCATION	L3	LOCATION			NF	1
HTL	1	L3	LOCATION					F	0.57

Figure C.3: Problem 2 Initial State

gpname	noargs	arg1	type1	arg2	type2	arg3	type3	ftype	mu	wt
UCD	1	U2	UNIT					F	0.89	1
UCD	1	U3	UNIT					F	0.856667	1
UCD	1	U4	UNIT					F	0.666667	1
PVNL	2	P1	PLANE	L5	LOCATION			F	0.98	1

Figure C.4: Problem 2 Goal State

Bibliography

[1] R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving in problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[2] M. Sarfraz and A. Raza. Visualization of data with spline fitting: A tool with genetic approach. *Proceedings of the International Conference on Imaging Science, Systems, and Technology (CISST'2002), Las Vegas, Nevada,USA, CSREA Press, USA.*, 2002.

[3] M. Sarfraz and S.A. Raza. Capturing outline of fonts using genetic algorithm and splines. *Proceedings of IEEE International Conference on Information Visualization-IV'2001-UK, IEEE Computer Society Press,USA*, pages 738–743, 2001.

[4] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, Englewood Cliffss, New Jersey 07632., 1 edition, 1995.

[5] G.J. Sussman. A computational model of skill acquisition. AI Technical Report 297, AI Labortaory, Massahusettes Institute of Technology, 1973.

[6] A. Tate. *Using Goal Structure to direct search in a problem solver.* PhD thesis, University of Edinburgh, September 1975.

[7] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[8] E. D. Sacerdoti. A structure of plans and behavior. *Tech. Note 109, AI Center, SRI International, Inc., Menlo Park, Calif*, 1975.

[9] M.J. Stefik. Planning with constraints. Technical Report 80-784, Computer Science Dept. Stanford University, 1980.

[10] B. Hayes Roth. Human planning processes. *Report No. R-2670-ONR, Rand Corporation., Santa Monica, California.*, 1980.

[11] E. Lehner Paul. *An introduction to knowledge-based planning: paradigms and approaches.* George Mason University, 1996.

[12] R. D. Luce and H. Raiffa. *Games and Decisions: Introduction and Critical Survey.* Wiley, 1957.

[13] D' B. Ambrosio. Inference in bayesian networks. *AI Magazine*, 20(2), 1999.

[14] L. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[15] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton NJ, 1976.

[16] K. Hammond. Case-based planning: A framework for planning from experience. *Journal of Cognitive Science*, 14(3), 1990.

[17] Z. K. Haigh and M. Veloso. Route planning by analogy. *Proceedings of 1st International Conference on Case-based Reasoning, Sesimbra, Portagal, Springer-Verlag*, pages 169–180, 1995.

[18] H. Munoz Avila. *Integrating twofold Case Retrieval and Complete Decision Replay in CAPLAN/CBC.* PhD thesis, University of Kaiserslautern, May 1998.

[19] M. Veloso. Planning and learning by analogical reasoning. Technical report, Number 886 in Lecture Notes in Artificial Intelligence. Springer Verlag, 1994.

[20] S. Kambhampati. Exploiting causal structure to control retrieval and refitting during plan reuse. *Computational Intelligence*, 10(2):213–244, 1994.

[21] J. Koehler. Flexible plan reuse in a formal framework. *In Current Trends in AI Planning, IOS Press, Amsterdam, Washington, Tokio*, pages 171–184, 1994.

[22] G. J. A. Francis and A. Ram. A domain independent algorithm for multiplan adaptation and merging in least commitment planning. *In Aha, D. and Rahm, A.,editors, AAAI Fall Symposium: Adaptation of Knowledge Reuse, Menlo Park, CA. AAAI Press*, 1995.

[23] R. Bergmann and W. Wilke. Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995.

[24] L. Ihrig and S. Kambhampati. Design and implementation of a replay framework based on a partial order planner. *In Weld, D., editor, Proceedings of AAAI96. IOS Press*, 1996.

[25] M. Veloso and J. Carbonell. Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Machine Learning*, 10, 1993.

[26] T. Mitchell, R. Keller, and S. KedarCabelli. Explanation based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[27] L. Ihrig and S. Kambhampati. Derivational replay for partial order planning. *In Proceedings of AAAI94*, pages 116–125, 1994.

[28] H. Munoz Avila, J. Paulokat, and S. Wess. Controlling nonlinear hierarchical planning by case replay. *Proceedings of the 2nd European Workshop on Case-Based Reasoning (EWCBR94*, 1994.

[29] M. Ahmed, E. Damiani, and D. Rine. Fast recall of reusable fuzzy plans using acyclic directed graph memory. *Proceedings of the 1998 ACM symposium on Applied Computing, Atlanta, GA USA*, pages 272–276, February 27 - March 1 1998.

[30] B. Nebel and J. Koehler. Plan reuse versus plan generation: a theoretical and empirical analysis. *Artificial Intelligence*, 76:427–454, 1995.

[31] B. Smyth and M. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102:249–293, 1998.

[32] B. Smyth and M.T. Keane. Experiments on adaptation-guided retrieval. *Case-Based Reasoning, Research and Development, M. Veloso and A. Aamodt (eds.), Lecture Notes in Artificial Intelligence 1010, Springer Verlag, Berlin*, pages 313–324, 1995.

[33] S. Kambhampati and J. A. Hendler. A validation structure based theory of plan modification and reuse. *Artificial Intelligence*, 55:192–258, 1992.

[34] S. Hanks and D. Weld. A domain independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research*, 2:319–360, 1995.

[35] K. Miyashita and K. Sycara. Cabins: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial Intelligence*, 76:377–426, 1995.

[36] K. Hammond. Chef: a model of case-based planning. *Proceedings of American Association of Artificial Intelligence, AAAI86*, 1986.

[37] J. Blythe, O. Etzioni, Y. Gil, R. Joseph, A. Perez, S. Reilly, M. Veloso, and X. Wang. Prodigy4.0: The manual and tutorial. *Technical report CMUCS92150, School of Computer Science, Carnegie Mellon University*, 1992.

[38] H. Munoz Avila, J. A. Hendler, and D. W. Aha. Conversational case-based planning. *New Review of Applied Expert Systems*, 5, 1999.

[39] H. Munoz Avila, D.W. Aha, L.A. Breslow, D.S. Nau, and R. Weber. Integrating conversational case retrieval with generative planning. *Proceedings of the Fifth European Workshop on Case Based Reasoning, Trento, Italy: Springer-Verlag*, pages 322–334, 2000.

[40] B. Falkeneheimer, D. K. Forbus, and D. Gentner. The structure mapping engine. *Proceeding of the Sixth National Conference on Artificial Intelligence, Philadelphia, PA, US*, 1986.

[41] D. Navichandra. Exploration and innovation in design towards a computational model. *Springer Verlag, New York. NY, US*, 1991.

[42] T. Acorn and S. Walden. Smart: Support management cultivated reasoning technology compaq customer service. *Proceedings of AAAI-92 . Cambridge. MA: AAAI Press/MIT Press*, 1992.

[43] E. Simoudis, A. Mendall, and P. Miller. Automated support for developing retrieve-and-propose systems. *Proceedings of Artificial Intelligence XI Conference, Orlando, Florida*, 1993.

[44] M.L. Maher and D.M. Zhang. Cadsyn: using case and decomposition knowledge for design synthesis. *Artificial Intelligence in Design, Gero, J.S. (ed.), Butterworth-Heinmann. Oxford. UK*, 1991.

[45] E. Domeshek. A case study of case indexing: Designing index feature sets to suit task demands and support parallelism. *Advances in connectionnist and neural computation theory, Analogical connections, eds. J. Barenden and K. Holyoak, Norwood, NJ. US*, 2, 1993.

[46] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic least-commitment planning. *Proceedings of Twelfth National Conference on Artificial Intelligence, AAAI Press*, pages 1073–1078, 1995.

[47] A. Barrett and D. S. Weld. Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112, 1994.

[48] P. Haddawy and M. Suwandi. Decision-theoretic refinement planning using inheritance abstraction. *In Hammond, K., ed., Proceedings of Second International Conference on AI Planning Systems. University of Chicago, Illinois: AAAI Press*, 1994.

[49] J. Blythe. *Planning Under Uncertainty in Dynamic Domains*. PhD thesis, Carnegie Mellon University, 1998.

[50] S. M. Majercik and M. L. Littman. Maxplan: A new approach to probabilistic planning. *Proceedings of the Fourth International Conference on Artificial Intelligence Planning*, 1998.

[51] D. E. Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley. Planning and reacting in uncertain and dynamic environments. *JETAI*, 7(1):197–227, 1995.

[52] H. Aisu, T. Unemi, Y. Inagaki, and H. Sugie. A planning architecture for intelligent robot: Fuzzy memory-based reasoning for real-time planning/control. *Industrial Electronic control , Instrumentation, Proceedings of IECON*, 1:165–169, 1993.

[53] H. Aisu, T. Unemi, Y. Inagaki, S. Ono, and H. Sugie. A robust planning and control system handling fuzziness. *Fuzzy Systems, 4th IEEE International Conference on Fuzzy Systems*, 3:1701–1706, 1995.

[54] E. H. Ruspini and D. Ruspini. Autonomous vehicle motion planning using fuzzy logic. *Proceedings of the IEEE Round Table on Fuzzy and Neural Systems and Vehicle Applications, Tokyo, Japan*, 1991.

[55] J. Yen and N. Pfluger. Path planning and execution using fuzzy logic. *AIAA Guidance, Navigation and Control Conference, New Orleans, LA*, 3:1691–1698, 1991.

[56] López de., R. Mántaras, and E. Plaza. Case-based reasoning: An overview. *AI Communications*, 10:21–29, 1997.

[57] E. Plaza and de Mantaras R. Lopez. A case-based apprentice that learns from fuzzy examples. *Methodologies for Intelligent Systems*, 5:420–427, 1990.

[58] M. Cayrol, H. Farreny, and H. Prade. Fuzzy pattern matching. *Kybernetes*, 11:103–116, 1982.

[59] S. Salotti. *Filtrage flou et reprèsentation centrèe objet pour raisonner par analogie: Le système FLORAN. (In French).* PhD thesis, University of Paris XI, Orsay, France., 1992.

[60] M. Jaczynski and B. Trousse. Fuzzy logic for the retrieval step of a case-based reasoner. *Proceedings of the European Workshop on Case-Based Reasoning, EWCBR-94*, pages 313–321, 1994.

[61] D. Dubois, H. Prade, and C. Testemale. Weighted fuzzy pattern matching. *Fuzzy sets and systems*, 28(3):313–331, 1988.

[62] K. B. Hansen. Weather prediction using case-based reasoning and fuzzy set theory. *Master of Computer Science Thesis, Technical University of Nova Scotia, Halifax, Nova Scotia, Canada*, 2000.

[63] J. M. Keller, M. R. Gray, and J. A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions. Systems, Man, and Cybernetics*, 15(4):258–263, 1985.

[64] T.Y. Slonim and M. Schneider. Design issues in fuzzy case-based reasoning. *Fuzzy Sets and Systems*, 117(2):251–267, Jan 2001.

[65] D. Medin and A. Ortony. *Comments on Part I: Psychological essentialism. In Similarity and Analogical Reasoning.* New York, NY. Cambridge University Press, 1989.

[66] D. Gentner. The mechanisms of analogical learning. *In Vosniadou, S. and Ortony, A., editors, Similarity and Analogical Reasoning, New York, NY, Cambridge University Press*, 199-241, 1989.

[67] L. J. Rips. Similarity, typicality and categorization. *Similarity and Analogical Reasoning, New York, NY. Cambridge University Press*, pages 21–59, 1989.

[68] S. Vosniadou. Analogical reasoning as a mechanism in knowledge acquisition: A developmental perspective. *Similarity and Analogical Reasoning, New York, NY. Cambridge University Press*, pages 413–437, 1989.

[69] B. H. Ross. Remindings in learning and instruction. *Similarity and Analogical Reasoning, New York, NY. Cambridge University Press*, pages 438–469, 1989.

[70] Jurisica Igor. Context-based similarity applied to retrieval of relevant cases. *Technical Report DKBS-TR-94-5, University of Toronto, Department of Computer Science, Toronto, Ontario*, 1994.

[71] J. Kolodner. *CaseBased Reasoning.* Morgan Kaufmann Publishers, 1993.

[72] T. Utsuro. Thesaurus-based efficient example retrieval by generating retrieval queries from similarities. *Proceedings of International Conference on Computational Linguistics*, pages 1044–1048, 1994.

[73] J.R. Quinlan. Induction over large databases. Heuristic Programming Project HPP-79-14, Computer Science Dept., Stanford University, US, 1979.

[74] Pews. Bergmann and Wilke. Explanation-based similarity: A unifying approach for integrating domain knowledge into case-based reasoning for diagnosis and planning tasks. lecture notes on artificial intelligence. *In S. Wess, K.-D. Althoff, and M.M. Richter, editors, Topics in Case-Based Reasoning*, 837:182–196, 1994.

[75] Pappis P. Costas and Karacapilidis I. Nikos. A comparative assessment of measures of similarity of fuzzy values. *Fuzzy Sets and Systems*, 56:171–174, 1993.

[76] Y. Wang, N. Inuzuka, and N. Ishii. A method of similarity metrics using fuzzy integration. *Proceedings of International Conference on AI (3rd Pacific)*, 2:1028–1034, 1994.

[77] M. S. Chen. A new approach to handling fuzzy decision-making problems. *IEEE Transaction. Systems, Man, Cybernetics.*, 18:1012–1016, Nov/Dec 1988.

[78] M. S. Chen. A weighted fuzzy reasoning algorithm for medical diagnosis. *Decision Support Systems.*, 11:37–43, 1994.

[79] D. S. Yeung. Improved fuzzy knowledge representation and rule evaluation using fuzzy petri nets and degree of subsethood. *Intelligent Systems*, 9(2):281–290, 1994.

[80] D. S. Yeung and E. C. Tsang. A comparative study on similarity-based fuzzy reasoning methods. *IEEE Transaction. Systems, Man, Cybernetics*, 27(2):216–227, April 1997.

[81] F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, MA, 1997.

[82] S.L. Saltezberg. A nearest hyper-rectangle learning method. *Machine Learning*, 6:251 –276, 1991.

[83] Wetterschereck and D. Aha. Weighting parameters, case-based reasoning research and development. *Proceedings of the 1st International Conference on Case-Based Reasoning, Springer-Verlag*, pages 347–358, 1995.

[84] H. Munoz Avila and J. Hullen. Feature weighting by explaining case based planning episodes, in lnai. springer. *Third European Workshop (EWCBR96)*, 1996.

[85] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms with Application in Engineering: Solving Combinatorial Optimization Problems.* IEEE Computer Society Press, California, 1999.

[86] Aldenderfer S. Marks and Blashfield K. Roger. Cluster analysyis. *Paper Series on Quantitative Applications in social sciences*, 07-044, 1984.

[87] S. Santini and R. Jain. Similarity matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(9):871 –883, Sept 1999.

[88] B.P. Kettler, J.A. Handler, W.A. Anderson, and M.P. Evett. Massively parallel support for case-based planning. *IEEE Expert*, pages 8–14, 1994.

[89] A. Tate, J. Hendler, and M. Drummond. A review of ai planning techniques. *In Readings in Planning, Allen J. Hendler. Tate A. (eds), San Francisco, CA: Morgan Kauffmann.*, 1990.

[90] G. D. Bridge. Defining and combining symmetric and asymmetric similarity measures. *Proceedings of the 4th European Workshop On Case-Based Reasoning*, pages 52–63, 1998.

[91] D. Gentner. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[92] J. Koehler. Planning from second principles. *Artificial Intelligence*, 87, Fall 1996.

[93] D. Leake. Constructive similarity assessment: Using stored cases to define new situations. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, Hillsadle, NJ: Lawrence Erlbaum*, pages 313–318, 1992.

[94] D. Leake. Adaptive similarity assessment for case-based explanation. *International Journal of Expert Systems*, 8(2):165–194, 1995.

[95] D. B. Leake. *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* AAAI Press/MIT Press, 1996.

[96] T. W. Liao, Z. Zhang, and C. R. Mount. Similarity measures for retrieval in case-based reasoning systems. *Applied Artificial Intelligence*, 12:267–288, 1998.

[97] D. Thomas. *Artificial Intelligence: Theory and Practice.* The Benjamin/Cumming Publishing Comp. Inc., 1995.

[98] Y. Wang and N. Ishii. A method of similarity metrics for structured representations. *Expert Systems with Application*, 12, 1997.

[99] Y. Wang and Ishii Naohiro. A genetic algorithm for learning weights in a similarity function. *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Norwich, UK, Springer-Verlag, Berlin*, pages 206–209, 1997.

Vita

- Malik Ahmed Owais.
- Born in Karachi, Pakistan on January 02, 1976.
- Received Bachelor of Engineering (B.E) degree in Computer Systems from N.E.D University of Engineering and Technology, Karachi, Pakistan in March 1998.
- Worked as a Lecturer in Usman Institute of Technology (Hamdard University) Karachi, Pakistan from April 1998 to December 1999.
- Joined the Department of Information and Computer Science at KFUPM as a Research Assistant in January 2000.
- Completed Master of Science (M.S.) in Information and Computer Science at KFUPM in May 2002.
- Email: malikowais@yahoo.com